On Semi-Online Machine Scheduling and Generalized Bin Covering

DISSERTATION

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Humboldt-Universität zu Berlin

von
Herrn Dipl.-Inf. Matthias Hellwig

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II:
Prof. Dr. Elmar Kulke

Gutachter:

1. Prof. Dr. Susanne Albers
2. Prof. Dr. Csanád Imreh
3. PD Dr. Alexander Souza

Tag der mündlichen Prüfung: 15.07.2013

Abstract

In this thesis we study algorithms for scheduling problems. We investigate semi-online minimum makespan scheduling and generalized bin covering. In online minimum makespan scheduling we are given a set of m machines and n jobs, where each job J_t is specified by a processing time. The jobs arrive one by one and we have to assign them to the machines without any knowledge about future incoming jobs. The load of a machine is defined to be total processing time of the assigned jobs. The goal is to place the jobs on the machines such that the maximum load of a machine is minimized. In semi-online minimum makespan scheduling this strict setting is softened. We investigate three different models. In the first setting an algorithm is given an advice on the total processing time of the jobs. We present a simple 1.75-competitive algorithm and a lower bound of ≈ 1.585. In the second setting we may reassign jobs upto a limited amount. In this model we present an algorithm that has a competitive ratio of $\rho_m \approx 1.4659$ for $m \to \infty$ and uses no more than $\varphi_m \cdot m$ job migrations, where φ_m is a constant between 7 and 10, depending on m. The result is complemented by two lower bounds. Firstly, no algorithm can attain a competitive ratio smaller than ρ_m using $o(n)$ migrations. Secondly, every algorithm that has competitiveness smaller than $1 + 1/\sqrt{2}$ must use $\Omega(m)$ job migrations. We finally trade performance for migrations and obtain a family of algorithms with competitiveness $\hat{\rho}$, for every $5/3 \leq \hat{\rho} \leq 1.75$, that uses between $4m$ and $2.5m$ job migrations.

The third semi-online setting we study is minimum makespan scheduling with parallel schedules. In this problem an algorithm may maintain several schedules, the best of which is output after the arrival of the entire job sequence. We provide a reduction to the special case, where the optimum solution value is known. We design an algorithm that maintains $O(1)$ schedules and is $(4/3 + \varepsilon)$-competitive, for any $\varepsilon > 0$. We furthermore give an algorithm that is $(1+\varepsilon)$-competitive, for any $\varepsilon > 0$, using $m^{O(1/\varepsilon)}$ schedules. Our results are complemented by a lower bound stating that $m^{\Omega(1/\varepsilon)}$ schedules have to be maintained in order to achieve a competitiveness of less than $1 + \varepsilon$, for every $\varepsilon \leq 1/3$.

In generalized bin covering we are given m bin types and n items. Each bin type M_j is specified by a demand d_j and a revenue r_j. Each item J_t has a size p_j. A bin of type M_j is said to be covered if the total size of the assigned items is at least the demand d_j. Then the revenue r_j is earned. The goal is to find an assignment of items to bins maximizing the total obtained revenue. We study two models of bin supply. In the unit supply model there is only one bin of each type available. By contrast in the infinite supply model each bin type is available arbitrarily often, and hence the former is a generalization of the latter. We give a 5-approximation for the unit supply model. In the special case that we have $d_j = r_j$, for all bin types M_j, we can give a $9/4$-approximation for the unit supply model. We show a lower bound on the approximation factor of 2, unless P = NP, that holds for the unit supply model. The result remains valid even asymptotically. For the infinite supply model we give an AFPTAS in the special case that $d_j = r_j$, for all bin types M_j.

Zusammenfassung

In dieser Arbeit untersuchen wir Algorithmen für Scheduling-Probleme. Wir betrachten semi-online Makespan-Scheduling und generalisiertes Bin Covering. Im online Makespan-Scheduling-Problem sind m Maschinen und n Jobs gegeben, wobei letztere jeweils eine individuelle Bearbeitungszeit haben. Es wird zu jedem Zeitpunkt ein Job offengelegt und muss sofort und unwiderruflich einer Maschine zugewiesen werden, ohne Wissen über zukünftige Jobs. Die Last einer Maschine wird als die Summe der Bearbeitungszeiten der ihr zugewiesenen Jobs definiert. Das Ziel ist es, eine Zuweisung von Jobs zu Maschinen zu finden, sodass die höchste Last einer Maschine minimiert wird. Im semi-online Scheduling-Modell wird dieses strikte Szenario relaxiert. Wir untersuchen drei verschiedene Modelle. Im ersten ist uns die kumulierte Bearbeitungszeit der Jobs vor Ankunft der einzelnen Jobs bekannt. Hierfür geben wir einen 1.75-kompetitiven Algorithmus sowie eine untere Schranke von ≈ 1.585. Im zweiten Modell dürfen wir bis zu einem gewissen Grade bereits zugewiesene Jobs anderen Maschinen neu zuordnen. Für dieses Modell geben wir einen Algorithmus mit einer Kompetitivität $\rho_m \approx 1.4659$, für $m \to \infty$, der nicht mehr als $\varphi_m \cdot m$ Neuzuweisungen vornimmt. Hierbei ist φ_m eine von m abhängige Konstante zwischen 7 und 10. Wir komplementieren dieses Ergebnis mit zwei unteren Schranken. Wir zeigen, dass kein Algorithmus mit einer Kompetitivität kleiner als ρ_m existiert, der nur $o(n)$ Neuzuweisungen durchführt. Weiterhin beweisen wir, dass jeder Algorithmus, der eine Kompetitivität kleiner als $1 + 1/\sqrt{2}$ hat, auch $\Omega(m)$ Neuzuweisungen vornehmen muss. Schließlich geben wir eine Familie von Algorithmen, die zugunsten weniger Neuzuweisungen eine schlechtere Kompetitivität hat. Die Kompetitivität $\hat{\rho}$ kann hier beliebig zwischen $5/3$ und 1.75 gewählt werden, bei nur $4m$ bis $2.5m$ Neuzuweisungen.

Im dritten semi-online Scheduling-Modell darf ein Algorithmus mehrere Lösungen parallel konstruieren, von denen die beste ausgegeben wird. Wir geben zunächst eine Reduktion des allgemeinen Problems auf den Spezialfall an, in dem der Wert einer optimalen Lösung bekannt ist. Wir benutzen diese Reduktion, um zwei Algorithmen zu entwickeln. Einer dieser Algorithmen ist $(4/3 + \varepsilon)$-kompetitiv, für jedes $\varepsilon > 0$, und konstruiert $O(1)$ Schedules. Der andere ist $(1 + \varepsilon)$-kompetitiv, für jedes $\varepsilon > 0$, und benötigt $m^{O(1/\varepsilon)}$ Schedules. Darüber hinaus geben wir folgendes Negativresultat. Jeder Algorithmus, der eine Kompetitivität kleiner als $1 + \varepsilon$ hat, für $0 < \varepsilon \leq 1/3$, muss mindestens $m^{\Omega(1/\varepsilon)}$ Schedules benutzen.

Beim generalisierten Bin Covering sind uns m Bintypen und n Objekte gegeben. Ein Bintyp M_j hat einen Bedarf d_j und einen Profit r_j. Jedes Objekt J_t hat eine Größe p_t. Ein Bin vom Typ M_j heißt abgedeckt (engl. „covered“), wenn die Summe der Größen der ihm zugewiesenen Objekte mindestens d_j ist. Wenn ein Bin vom Typ M_j abgedeckt ist, erzielen wir einen Profit von r_j. Ziel ist es, die Objekte Bins zuzuweisen, sodass der erzielte Gesamtprofit maximiert wird. Wir untersuchen zwei Modelle, die sich in der Verfügbarkeit von Bintypen unterscheiden. Im Unit-Supply-Modell steht uns von

jedem Bintyp genau ein Bin zur Verfügung. Im Gegensatz dazu stehen uns im Infinite-Supply-Modell von jedem Bintyp beliebig viele Bins zur Verfügung. Das Unit-Supply-Modell ist daher eine Verallgemeinerung des Infinite-Supply-Modells. Unsere Resultate umfassen eine 5-Approximation im Unit-Supply-Modell und eine $9/4$-Approximation für den Spezialfall, dass $r_j = d_j$ für alle Bintypen gilt, ebenfalls im Unit-Supply-Modell. Wir zeigen unter der Annahme P $\neq$ NP, dass kein deterministischer Algorithmus mit polynomieller Laufzeit einen besseren Approximationsfaktor als 2 erzielen kann. Dieses Ergebnis gilt auch asymptotisch. Im Spezialfall, dass $r_j = d_j$ für alle Bintypen gilt, zeigen wir für das Infinite-Supply-Modell die Existenz eines AFPTASes.

Dedicated to my family

Acknowledgements

First I would like to thank my advisor Prof. Dr. Susanne Albers for employing me as a member of her working group, and thus giving me the opportunity to work on this thesis. I am grateful for her academic advice/supervision. Her proposals of research problems were very valuable and led to fruitful work. I also appreciate that I could be a coauthor of hers. We could obtain several joint results, and thanks to her collaboration the presentation of many of them is now the way it is.

For similar reasons I would like to thank PD Dr. Alexander Souza, with whom I could publish a paper, too. I acknowledge his advice that was valuable as well and also led to nice joint work.

Even though we have not (yet) obtained publishable results, I would also like to thank Prof. Dr. Csanád Imreh for the good collaboration during his stay in Berlin.

I thank all my colleagues who I have not listed so far. I believe that I have learned a lot from the stimulating discussions with them about problems in both research and teaching. In order of first meeting them: My long-term room mate Dr. Antonios Antoniadis, Pascal Lenzner, Carsten Moldenhauer, Matthias Killat, Achim Passen and Dr. Chien-Chung Huang.

For their strong organizational help I would like to thank Ralf Oelschlaegel and Eva Sandig.

Finally, my dear spouse Elmira, I am deeply grateful for your everlasting encouragement and understanding. My family's and your support were a great aid to me throughout the difficult phases.

Contents

Chapter 1

Introduction

1.1 Motivation and Scope

Scheduling is a field of algorithm design with high practical relevance. In a scheduling problem we are given a set of resources and a set of tasks. Each task can be assigned to a resource. Equivalently, this can be seen as assigning each resource to a subset of the tasks, and for distinct resources the subsets of assigned tasks have to be disjoint. The assignments have to be subject to a certain set of constraints, for example that every task be assigned, and we have to maximize or to minimize an objective.

In practice there is a vast amount of scheduling problems, for example stemming from fields as logistics or computer science. Several of these problems are highly non-trivial, which results in a need for designing efficient algorithms. Maybe one of the most important scheduling problems in computer science, which we also study in this thesis, is the minimum makespan problem. In this problem resources are also called machines and tasks are called jobs. The problem is informally defined as follows. Jobs, which each have a certain processing time, have to be assigned to machines such that the maximum load on the machines is minimal. Here, the load of a machine is the total processing time of the jobs assigned to that machine.

In scheduling three main models have to be distinguished. In offline scheduling all the information about the problem is known to the algorithm in advance. This may be an unrealistic assumption in practice since information about the problem may only be delivered incrementally. In online scheduling nothing is known about the input in advance and jobs are revealed one by one. Decisions have to be made irrevocably without any knowledge about the future. A hybrid of the both models are semi-online models, and the respective algorithms therein are called semi-online algorithms. Often problems are neither pure online nor pure

offline in practice. This motivates semi-online models in which the pure online setting is relaxed, in one or several ways. For example, semi-online algorithms can be provided with a piece of additional information, also called an advice, or the strict requirement of scheduling jobs irrevocably is relaxed up to a certain limit.

In this thesis we investigate scheduling in offline and semi-online scenarios. In the next section we formally describe scheduling problems, we introduce the models in which they are considered, and we also give the means for evaluating the performance of algorithms in the respective models. Moreover, the basic notation is already introduced. Thereafter we define the problems considered in this thesis and give a brief overview of our results.

1.2 Preliminaries

1.2.1 Scheduling Problems

In this thesis we describe a scheduling problem by the following components. An *instance* is a pair (μ, σ) consisting of a set $\mu = \{M_1, \ldots, M_m\}$ of m *resources* and a set $\sigma = \{J_1, \ldots, J_n\}$ of n *tasks*. Depending on the problem setting we study, resources and tasks may have properties in addition. In this case these properties have to be specified by an instance. To summarize, an instance (μ, σ) describes the input for an algorithm. Algorithms are denoted by small caps in this thesis, for example we refer with ALG to an algorithm called "ALG". In particular, OPT always denotes an optimal algorithm for the given problem. Every algorithm should output a *solution* $S = (S_1, \ldots, S_m)$ to the input instance (μ, σ). We require each S_j to be a subset $S_j \subseteq \sigma$ and the S_j to be disjoint, i. e. $S_j \cap S_{j'} = \emptyset$ for $j \neq j'$. Here S_j is the set of tasks that are assigned to resource M_j. In classical scheduling problems solutions are also referred to as *schedules*.

The goal is either to *minimize* or to *maximize* a certain *objective (function)* f subject to a set of *constraints*. The function f maps each solution S to a real number $f(S) \in \mathbb{R}$, and $f(S)$ is called the *(objective) value (of the solution S)*. Intuitively, the function f evaluates the quality of a solution. Throughout this thesis we denote the value of a solution given by an algorithm ALG to an instance (μ, σ) by $\text{ALG}(\mu, \sigma)$. If no confusion arises we may also omit the instance and simply denote the objective value by ALG. Consequently, the value of an optimal solution is referred to as $\text{OPT}(\mu, \sigma)$ or simply OPT. Depending on the problem, we are either to minimize or to maximize the (objective) value of a solution in order to provide a solution of high quality. The constraints define the set of *feasible* solutions among the set of all solutions. They require solutions to have certain properties and thus restrict the set of solutions an algorithm may output. Typically

we refer to a feasible solution simply as a "solution". Only in certain contexts in Chapter 5 infeasible solutions are interesting to us, which we have then to transform into a feasible solution before our algorithms may give an output.

A *performance measure* evaluates the performance guarantee of an algorithm. Most commonly, a performance measure compares the *worst-case* behavior of an algorithm ALG to an optimal offline algorithm OPT on the set of all instances, and we devise *worst-case analysis* in this thesis exclusively. Performance measures depend on the *setting* (or *model*), and we define several performance measures below. We study scheduling problems in three different settings: In the *offline model*, in the *online model* and in *semi-online models*. Each setting describes which kind of information is available to an algorithm, possibly defines a certain set of constraints and allows or disallows a certain set of actions an algorithm may perform or may not perform in order to provide a solution.

Offline Scheduling

In the offline setting the whole input is known to an algorithm before computation starts. We assume that the reader is familiar with the notions of *decision/optimization problems*, *Turing machines*, *algorithms* and *reductions*, see [53, 57] for an introduction. A decision problem L is contained in the complexity class P if there exists a deterministic Turing machine M with polynomial running-time such that M accepts every word x if and only if $x \in L$. A decision problem L is contained in the complexity class NP if there exists a non-deterministic Turing machine M with polynomial running-time such that M accepts each word x if and only if $x \in L$. For problems L, L' let $L' \leq_p L$ denote that L' is reducible to L in polynomial time. A problem L is NP-hard if for all $L' \in$ NP there holds $L' \leq_p L$. For a deeper introduction to NP-completeness theory see [53, 57]. It is commonly assumed that P $\neq$ NP. Under this assumption there exist no deterministic algorithms with polynomial running-time that solve NP-hard problems. It follows that optimization problems whose decision variants are NP-hard are not solvable in polynomial-time if P $\neq$ NP. Nonetheless, there may be algorithms that approximate a solution to an optimization problem in polynomial-time.

To evaluate an algorithm that finds approximate solutions to a problem whose decision variant is NP-hard, it is common to use the performance measures of *approximation ratio* and *asymptotic approximation ratio*. We remark that there exist different definitions of (asymptotic) approximation ratios depending whether maximization or minimization problems are considered [56]. We consider offline scheduling only for a maximization problem, thus we can use the following notions for simplicity reasons. The approximation ratio $\rho(\text{ALG})$ of an algorithm ALG is defined as

$$\rho(\mathrm{ALG}) = \sup_{(\mu,\sigma)} \frac{\mathrm{OPT}(\mu,\sigma)}{\mathrm{ALG}(\mu,\sigma)},$$

where the supremum is taken over the set of all instances (μ, σ).

We define the asymptotic approximation ratio $\overline{\rho}(\mathrm{ALG})$ of an algorithm ALG for a maximization problem to be

$$\overline{\rho}(\mathrm{ALG}) = \lim_{r\to\infty} \sup_{\substack{(\mu,\sigma),\\ \mathrm{OPT}(\mu,\sigma)\geq r}} \frac{\mathrm{OPT}(\mu,\sigma)}{\mathrm{ALG}(\mu,\sigma)},$$

where again the supremum is taken over the set of all instances (μ, σ) that admit a solution of value at least r. We remark that there are different definitions for the asymptotic approximation ratio, for example depending on the instance size, which is typically $|\sigma|$ or $|\mu| + |\sigma|$. However, as we explain in Section 5.3.2, such a definition is not reasonable in our setting.

Algorithms with polynomial running-time that have certain performance guarantees are called *approximation algorithms*. If $\rho(\mathrm{ALG}) \leq \rho$ holds for an algorithm ALG with polynomial running-time in the input length, then algorithm ALG is called a *ρ-approximation*. If there is a family of algorithms with polynomial running-time such that for every $\varepsilon > 0$ the family contains an algorithm with approximation guarantee $1 + \varepsilon$, then the respective family of algorithms is called a *polynomial-time approximation scheme* or *PTAS* for short. If the running-time of the algorithms is additionally polynomial in $1/\varepsilon$, then the respective family is called a *fully polynomial-time approximation scheme* or *FPTAS* for short.

If an algorithm with polynomial running-time has an asymptotic approximation ratio, then this algorithm is called an *asymptotic approximation algorithm*. The notions of *asymptotic polynomial-time approximation scheme (APTAS)* and *asymptotic fully polynomial-time approximation scheme (AFPTAS)* transfer analogously. We study an offline scheduling problem in Chapter 5.

Online Scheduling

In this thesis we consider in online scheduling the minimum makespan problem only. Thus, in online scheduling, we refer to the set μ of an instance (μ, σ) as the set of machines and to σ as the set of jobs, as this is common usage.

In the online scheduling problem the set of machines μ is known to an algorithm in advance. The set σ is considered to be (an ordered) sequence $\sigma = J_1, \ldots, J_n$. At each *time* t, $1 \leq t \leq n$, the job J_t is revealed and it has to be assigned irrevocably to a machine $M_j \in \mu$ without any knowledge about the future jobs $J_{t+1}, \ldots, J_n$.

An algorithm that obeys this model is a called an *online algorithm*. To evaluate the performance guarantee of an online algorithm the performance measure of *competitiveness* was introduced by Sleator and Tarjan [54]. We define the *competitive ratio* $\rho(\text{ALG})$ of an online algorithm ALG to be

$$\rho(\text{ALG}) = \sup_{(\mu,\sigma)} \frac{\text{ALG}(\mu,\sigma)}{\text{OPT}(\mu,\sigma)},$$

where again (μ, σ) is taken from the set of all instances. Note that $\text{OPT}(\mu, \sigma)$ here still refers to an optimal offline algorithm, whereas ALG is an online algorithm. An online algorithm ALG with $\rho(\text{ALG}) \leq \rho$ is said to be ρ-competitive. We remark that it is desired that the running-time of an online algorithm ALG is bounded polynomially in the input size but this is not strictly required. For a deeper introduction to online scheduling and competitive analysis we refer the reader to [16].

Semi-Online Scheduling

Again, as we study in semi-online scheduling the minimum makespan problem only, we refer to the resources as machines and to the tasks as jobs in this section only. Moreover we call solutions schedules. In practice problems are often neither pure offline nor pure online. This motivates the study of semi-online models, which are a hybrid of the online and the offline model. In *semi-online scheduling* the strict online setting is relaxed in one or several ways. The respective algorithms in this model are called *semi-online algorithms*. We give a comprehensive overview of the semi-online models studied in this thesis. In practice sometimes additional information about the instance is available though its exact appearance is unknown in advance. For example certain properties of the jobs in σ could be known and given to an algorithm, see e. g. [7, 9]. An online algorithm that is provided a limited amount of information about the set σ is said to get an *advice*. We study semi-online scheduling with advice in Chapter 2.

Sometimes the requirement that jobs be assigned irrevocably to machines is softened. Instead a limited number of reassignments of already assigned jobs is admitted [4, 49, 55]. We study semi-online scheduling with job migration in Chapter 3.

The performance guarantee of semi-online algorithms of the above types is evaluated using the measure of competitiveness, where, as usual, the semi-online algorithm is compared to an optimal offline algorithm. There are also semi-online models in which the performance measure is varied. For example, instead of comparing an online algorithm that maintains only one schedule S on an instance (μ, σ) to an optimal offline algorithm, the online algorithm may compute a set $\mathcal{S} = \{S^{(1)}, \ldots, S^{(k)}\}$ of schedules for one job sequence in parallel, i. e. each job

J_t is sequenced in each schedule $S^{(l)}$. The best of these schedules in $\mathcal{S}$ is compared to an optimal schedule. This model was introduced by Kellerer et al. [45]. Formally, we can define the performance measure as follows. Assume $\mathcal{S}(\mu, \sigma)$ is the set of schedules created by an algorithm ALG on the instance (μ, σ). Then the competitive ratio $\rho(\text{ALG})$ of ALG is

$$\rho(\text{ALG}) = \sup_{(\mu,\sigma)} \frac{\min_{S \in \mathcal{S}(\mu,\sigma)} f(S)}{\text{OPT}(\mu, \sigma)},$$

where f is the objective function and the supremum is taken over the set of all instances. As usual ALG is said to be ρ-competitive if $\rho(\text{ALG}) \leq \rho$.

There is an alternative view on this setting. One can think of a semi-online algorithm ALG in this model being a family of algorithms, i. e. $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$, where each online algorithm ALG_l maintains its own schedule $S^{(l)}$. Here I denotes some (finite) index set. From this perspective the family $\{\text{ALG}_l\}_{l \in I}$ competes with an optimal offline algorithm OPT. A family of algorithms $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$ has competitive ratio $\rho(\text{ALG})$ if

$$\rho(\text{ALG}) = \sup_{(\mu,\sigma)} \frac{\min_{l \in I} \text{ALG}_l(\mu, \sigma)}{\text{OPT}(\mu, \sigma)}.$$

Intuitively, for each instance (μ, σ) we compare the best algorithm ALG_l from the family to the optimal algorithm OPT. Again we call the family ALG ρ-competitive if $\rho(\text{ALG}) \leq \rho$. A semi-online problem with this performance measure is studied in Chapter 4.

We give more practical motivation for the above semi-online models in Section 1.3.1, in which we introduce the problems we consider in the respective models. In general, let us mention that semi-online scheduling problems are also thrilling from a theoretical point of view for several reasons. We think it is of deep theoretical interest which kind of (limited) additional information or which kind of (limited) additional power that an algorithm is provided with achieves a considerable improvement in terms of performance guarantee. Moreover, considering semi-online problems as special cases of pure online problems leads to a deeper understanding of problems that are not yet well-understood. Hence investigating semi-online problems can contribute to eventually solving difficult pure online problems.

1.2.2 Linear Programs

Linear Programming, also known as linear optimization, is a fruitful field of mathematical research and provides a powerful toolkit for solving optimization problems. Mathematical optimization problems are described in terms of *(integer)*

linear programs. A linear program in n variables, $x_1, \ldots, x_n$, with m constraints is described as

$$\begin{array}{lllllllll} \text{maximize} & c_1x_1 & + & \ldots & + & c_nx_n & & & \\ & & & & & & & & \\ \text{subject to} & a_{1,1}x_1 & + & a_{1,2}x_2 & + & \ldots & + & a_{1,n}x_n & \leq & b_1 \\ & & & & & \vdots & & & & \vdots \\ & a_{m,1}x_1 & + & a_{m,2}x_2 & + & \ldots & + & a_{m,n}x_n & \leq & b_m \\ & & & & & & & & & \\ & & & & & & & x_1 & \geq & 0 \\ & & & & & & & \vdots & & \vdots \\ & & & & & & & x_n & \geq & 0 \end{array} \tag{1.1}$$

where $a_{j,i}, c_i, b_j \in \mathbb{R}$, for all $1 \leq i \leq n$ and $1 \leq j \leq m$.

The function $c_1x_1 + \cdots + c_nx_n$, which is maximized, is called the *objective function*, the inequalities of the shape $a_{j,1}x_1 + \cdots + a_{j,n}x_n \leq b_j$ are called the *constraints* and the inequalities $x_i \geq 0$ are called the *non-negativity constraints* or *non-negativity conditions*.

For a vector x let $x^\top$ denote its transpose. A linear program can be formulated with a matrix $A = (a_{j,i})_{1 \leq j \leq m, 1 \leq i \leq n} \in \mathbb{R}^{m \times n}$ and vectors $c = (c_1, \ldots, c_n)^\top \in \mathbb{R}^n$, $b = (b_1, \ldots, b_m)^\top \in \mathbb{R}^m$. Let $x = (x_1, \ldots, x_n)^\top$, then the problem (1.1) can be formulated as

$$\max\{c^\top x \mid Ax \leq b, x \geq 0\}, \tag{1.2}$$

and formulation (1.2) is called the *canonical form* (of a linear program). Here, A is called the *coefficient matrix* and $c^\top x$ is the *objective function*. A vector $x \in \mathbb{R}^n$ is called a *solution*, and a solution is said to be *feasible* if it satisfies the constraints, i. e. if there holds $Ax \leq b$ and $x \geq 0$. Notice in particular that the formulations (1.1) and (1.2) are equivalent. Also, a linear program can be rewritten such that the objective function is minimized instead of maximized. We remark that finding an optimal solution to a linear program is a problem solvable in polynomial time, which was first shown by Khachiyan [46].

If the non-negativity constraints are replaced by *integrality constraints*, i. e. it is required that $x_i \in \mathbb{Z}$ for $1 \leq i \leq n$, linear programming becomes NP-hard, which can be seen easily. A linear program containing integrality constraints for all variables is called *integer linear program*. For a deeper introduction to (integer) linear programming we refer the reader to [50].

1.3 Overview

In this section we give a summary of the problems studied in this thesis. They fall into two research areas: minimum makespan scheduling and bin covering. We also give a brief overview of our results.

1.3.1 Minimum Makespan Scheduling

In Chapters 2, 3 and 4 we study the ONLINE MINIMUM MAKESPAN SCHEDULING problem. Makespan minimization on m identical machines is a fundamental scheduling problem. We are given a set $\mu = \{M_1, \ldots, M_m\}$ of m identical machines and a sequence $\sigma = J_1, \ldots, J_n$ of n jobs, where each job J_t is specified by a processing time p_t. Recall that in ONLINE MINIMUM MAKESPAN SCHEDULING at each time t the job J_t is presented and has to be assigned to a machine M_j without any knowledge of the future jobs $J_{t+1}, \ldots, J_n$.

Remember that a solution $S = (S_1, \ldots, S_m)$ describing the assignment of jobs to machines is called a *schedule*. Here, S_j denotes the set of jobs assigned to machine M_j. We will further call every partial solution in which only the jobs $J_1, \ldots, J_t$, with $t \leq n$, are assigned to their machines a schedule. With $\ell(j,t)$ we denote the total processing time of the jobs in $J_1, \ldots, J_{t-1}$ that are assigned to machine M_j, and $\ell(j,t)$ is called the *load* of machine M_j at time t. Formally, we define $\ell(j,t) = \sum_{t'<t:J_{t'} \in S_j} p_{t'}$, where S_j possibly refers here to a partial solution. If the time t is clear from the context, we write $\ell(j)$ for short to denote the load of machine M_j at the respective point in time. The *makespan* of a schedule at time t is the maximum load of a machine at time t, that is $\max_{1 \leq j \leq m} \ell(j,t)$. The goal in minimum makespan scheduling is to minimize the makespan at time $n+1$ subject to every job $J_1, \ldots, J_n$ being assigned to some machine.

For the classical MINIMUM MAKESPAN SCHEDULING problem in the online model Graham gave the first deterministic algorithm in 1966 [34]. He showed that the famous LISTSCHEDULING algorithm is $(2 - \frac{1}{m})$-competitive. Using new strategies, the competitive ratio was improved to $(2 - \frac{1}{m} - \varepsilon_m)$ [32], where $\varepsilon_m \to 0$ as $m \to \infty$, then to 1.986 [14] and 1.945 [44], and finally to 1.923 [2] and 1.9201 [31]. As for lower bounds, Faigle, Kern and Turan [28] showed that no deterministic online algorithm can achieve a competitiveness smaller than $2 - \frac{1}{m}$, for $m = 2$ and $m = 3$. For $m = 4$, Rudin and Chandrasekaran [41] gave a lower bound of $\sqrt{3} \approx 1.732$. For general m the lower bound was raised from 1.707 [28] to 1.837 [15] and 1.852 [2], and finally to 1.854 [33] and 1.88 [40].

For sake of completeness we also mention results in the online model using randomization. Randomized algorithms may use coin flips in order to make decisions. The makespan of the schedule of an randomized algorithm on a fixed instance is a random variable and hence one compares the expected makespan to

the optimum makespan to determine the competitiveness of an algorithm. For randomized online algorithms there exists for larger m a significant gap between the best known upper and lower bounds. For $m = 2$ machines, Bartal et al. [14] presented an algorithm that achieves an optimal competitive ratio of 4/3. Seiden [51] gave a randomized algorithm whose competitive ratio is smaller than the best known deterministic ratio for $m \in \{3, \ldots, 7\}$. For general m randomized online algorithms can not achieve a competitive ratio smaller than $e/(e-1) \approx 1.58$ [18, 52]. Let us remark that for arbitrary m no randomized algorithm whose competitive ratio is provably below the deterministic lower bound is currently known. The best known upper bound is a 1.916-competitive algorithm, which was devised in [3].

This relatively high competitiveness and the lack of progress in the area of randomized online strategies have led recent research to investigate scenarios where the online setting is relaxed. Furthermore, as already mentioned, in practice problems are often neither pure offline nor pure online. In this thesis we study three different semi-online settings, each of which has a natural motivation.

In Chapter 2 we study semi-online scheduling, where an online algorithm knows the total processing time $p^+ = \sum_{t:J_t \in \sigma} p_t$ of the jobs in the sequence σ. We believe that knowing p^+ is a very mild form of advice. We make no assumptions regarding the processing times of individual jobs and generally do not restrict the family of allowed job sequences. Availability of advice p^+ is also motivated by practical applications. In a parallel server system there usually exist fairly accurate estimates of the workload that arrives over a given time horizon.

In this chapter we make two contributions for semi-online scheduling with advice. We present a new semi-online algorithm that is based on an approach different from that of previous strategies. The algorithm is 1.75-competitive and does not achieve the best possible competitiveness. However, our algorithm is extremely simple and, unlike previous strategies, does not resort to job classes. The algorithm is more in the spirit of online algorithms not using any extra information. Hence our upper bound highlights the additional power of a small piece of advice provided to an online algorithm. We develop an improved lower bound showing that no deterministic semi-online algorithm can attain a competitive ratio smaller than 1.585. This reduces the gap between the previously known lower bound of 1.565 [45] and the upper bound of 1.6 [19].

In Chapter 3 we explore the power of job migration. In this setting an online scheduler is allowed to perform a limited number of job reassignments. A job that has already been assigned to some machine at a former time may be placed on a different machine. These reassignments may be performed at each time during the assignment of the jobs. Migration is a common technique used in theory and practice to balance load in parallel processing environments. It leads to improved processor utilization and reduced processing delays. Migration policies have been

analyzed extensively in theory and practice.

The model was introduced by Tan and Yu [55] giving an optimal algorithm for the special case of $m = 2$ machines. As our main result we settle the performance that can be achieved by deterministic online algorithms, for all m. We develop an algorithm that is ρ_m-competitive, for any $m \geq 2$, where ρ_m is the solution of a certain equation. For $m = 2$, $\rho_2 = 4/3$ and $\lim_{m\to\infty} \rho_m = W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2)) \approx 1.4659$. Here W_{-1} is the lower branch of the Lambert W function. For $m \geq 11$, the algorithm uses at most $7m$ migration operations. For smaller m, $8m$ to $10m$ operations may be performed. We complement this result by the following matching lower bounds: No online algorithm that uses $o(n)$ job migrations can achieve a competitive ratio smaller than ρ_m. Furthermore, a competitiveness of smaller than $1 + 1/\sqrt{2} > 1.707$ can not be achieved with $o(m)$ migrations. We finally trade performance for migrations. We give a family of algorithms that is $\hat{\rho}$-competitive, for any $5/3 \leq \hat{\rho} \leq 2$. For $\hat{\rho} = 5/3$, the strategy uses at most $4m$ job migrations. For $\hat{\rho} = 1.75$, at most $2.5m$ migrations are used.

Our third contribution in Chapter 4 studies ONLINE MAKESPAN MINIMIZATION WITH PARALLEL SCHEDULES. In this problem an online algorithm ALG is allowed to maintain a set $\mathcal{S} = \{S^{(1)}, \ldots, S^{(k)}\}$ of several schedules in parallel while processing the input sequence σ. At the end of the scheduling process the best schedule in $\mathcal{S}$ is selected. This model can be viewed as providing an online algorithm with extra space, which is invested to create multiple solutions.

To the best of our knowledge, very little is known about the value of extra space in the design of online algorithms. Makespan minimization with parallel schedules is of particular interest in processing environments where each processor can take care of a single or a small set of schedules. In fact we develop algorithms that require hardly any coordination or communication among the schedules/processors. The approach to grant an online algorithm extra space, invested to maintain multiple solutions, could be interesting in other problems as well.

We present a reduction of the problem to the special case where the optimum makespan is known. For this the number of schedules needed by an algorithm for the special case is multiplied by a constant depending on ε. Using this reduction we develop as our main result a $(4/3 + \varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, which uses a constant number of schedules. The constant is equal to $1/\varepsilon^{O(\log(1/\varepsilon))}$. We also give a $(1 + \varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, which builds a number of $(m/\varepsilon)^{O(\log(1/\varepsilon)/\varepsilon)}$ schedules. This value depends on m but is independent of the input sequence σ. The performance guarantees are nearly best possible. We show that any algorithm that achieves a competitiveness smaller than $1 + \varepsilon$ must construct $m^{\Omega(1/\varepsilon)}$ schedules, for every $0 < \varepsilon \leq 1/3$. For the specific value of $\varepsilon = 1/3$ we give a better bound. We prove that every $4/3$-competitive algorithm must construct at least $\lfloor m/3 \rfloor + 1$ schedules.

1.3.2 Generalized Bin Covering

In classical BIN COVERING we have an infinite supply of unit-sized bins and a collection of items having individual sizes. The objective is to pack items into as *many* bins as possible. That is, we seek to maximize the number of *covered* bins, where a bin is covered if the total size of the assigned items is at least the size of the bin. This problem is the dual of the classical BIN PACKING problem, where the goal is to pack the items into as *few* bins as possible; see the survey [20]. In this thesis we study GENERALIZED BIN COVERING: We have a set $\mu = \{M_1, \ldots, M_m\}$ of *bin types*, and each bin type $M_j \in \mu$ has a *revenue* r_j and a *demand* d_j. We denote the set of items by $\sigma = \{J_1, \ldots, J_n\}$ and define that each item $J_t \in \sigma$ has a *size* p_t. A bin of type M_j is *covered* or *filled* if the total size of the assigned items is at least the demand d_j of the bin, in which case we earn revenue r_j. The goal is to find a solution that maximizes the total obtained revenue. The special case with $r_j = d_j$, for each $1 \leq j \leq m$, is known as VARIABLE-SIZED BIN COVERING. The special case with $r_j = d_j = 1$, for each $1 \leq j \leq m$, is the classical BIN COVERING problem. To the best of our knowledge, the model with general revenues and demands has not been studied in the BIN COVERING setting before. We consider two models regarding the supply of bins: In the *infinite supply model* – as the name suggests – we have arbitrarily many bins available of each bin type. By contrast, we introduce the *unit supply model*, in which we have exactly one bin per type available. Hence in this model we rather speak of individual bins than of bin types. Note that these individual bins are allowed to have identical demands and identical revenues. It is not hard too see that the unit supply model is more general than the infinite supply model: By introducing n copies of each bin type, we can simulate the infinite supply model with the unit supply model. This changes the size of an instance only polynomially. A simulation of the unit supply model by the infinite supply model is obviously not possible. Hence our newly introduced model is more general than the existing infinite supply model. By a straightforward reduction (cf. Section 5.3.2) it is not hard to see that BIN COVERING is NP-hard in both models and can not be approximated better than 2, unless P = NP. We study the GENERALIZED BIN COVERING problem in the offline model in Chapter 5.

For motivating these generalizations, we mention the following application from trucking. Suppose that a moving company receives a collection of inquiries for moving contracts. Each inquiry has a certain volume and yields a certain revenue if it is served entirely. The company has a fleet of trucks where each truck has a certain capacity. The objective is to decide which inquiries to serve with the available trucks as to maximize total revenue. Inquiries are mapped to bins, whereas trucks are mapped to items in the GENERALIZED BIN COVERING setting. Notice that in particular the unit supply model is essential here since the

inquiries are individual, i. e. are not available arbitrarily often. All previous work on BIN COVERING exclusively considers the infinite supply model and hence is not applicable here. Also note that the GENERALIZED BIN COVERING problem applies in particular if the revenues do not necessarily correlate with the volume but also depend on the types of goods.

Our results in the unit supply model hold not only asymptotically but for all instances. This contrasts most of the previous work on BIN COVERING, which has been asymptotic. We prove that there is a combinatorial 5-approximation algorithm for GENERALIZED BIN COVERING with unit supply, which has running time $O(nm\sqrt{m+n})$, where m is the number of bins and n is the number of items in the instance.

Furthermore, we show that the natural and fast NEXT FIT DECREASING algorithm is a $9/4$-approximation in the unit supply model for VARIABLE-SIZED BIN COVERING. We show that our analysis of the algorithm is tight. Note that the approximation guarantee of the algorithm is not far away from the lower bound of two, assuming P $\neq$ NP.

Moreover, we provide an investigation of asymptotics in the unit supply model. Since bins are not available arbitrarily often in this model, there are problems for defining a meaningful asymptotics therein. We elaborate more on this issue in Section 5.3.2.

Finally, we give an AFPTAS for VARIABLE-SIZED BIN COVERING in the *infinite* supply model, which is obtained by extending existing A(F)PTASes for the classical BIN COVERING problem.

1.4 Listing of Publications

This thesis is based on the following publications. In brackets we append which chapters of this thesis are based on the respective publications.

- Susanne Albers and Matthias Hellwig. Semi-Online Scheduling Revisited. *Theoretical Computer Science*, 443:1–9. 2012. (Chapter 2).
- Susanne Albers and Matthias Hellwig. On the Value of Job Migration in Online Makespan Minimization. *Proc. 20th Annual European Symposium on Algorithms.* 84–95. 2012. (Chapter 3).
- Susanne Albers and Matthias Hellwig. Online Makespan Minimization with Parallel Schedules. *Submitted*. 2013. (Chapter 4).
- Matthias Hellwig and Alexander Souza. Approximation Algorithms for Generalized and Variable-Sized Bin Covering. *Proc. 15th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems.* 194–205. 2012. (Chapter 5).

Chapter 2

Semi-Online Scheduling Revisited

In this chapter we investigate basic online makespan minimization assuming that, additionally, the sum $p^+ = \sum_{t=1}^{n} p_t$ of the jobs' processing times is known.

2.1 Introduction

The Model In this chapter we consider basic online makespan minimization with advice. Recall that we denote the set of m machines with $\mu = \{M_1, \ldots, M_m\}$ and let the sequence of n jobs be $\sigma = J_1, \ldots, J_n$. Remember that at each time t the job J_t is presented and has to be assigned irrevocably to some machine from μ. The only information known about future jobs is their total processing time p^+.

Previous Work Besides the literature already mentioned in Section 1.3.1 there are several papers relevant to our work. The setting was first introduced by Kellerer et al. [45] who concentrated on $m = 2$ machines and gave a deterministic semi-online algorithm, which achieves an optimal competitive ratio of $4/3$. Again for $m = 2$, two papers by Angelelli et al. [8, 9] refined the results assuming that, additionally, the job processing times are upper bounded by a known value. A setting with $m = 2$ uniform machines was studied in [10].

Semi-online scheduling on a general number m of identical machines was investigated by Angelelli, Nagy, Speranza and Tuza [7] and Cheng, Kellerer and Kotov [19]. The studies must have been done independently since none of the two papers cites the other one. Angelelli et al. [7] gave a deterministic semi-online algorithm that attains a competitiveness of $(1+\sqrt{6})/2 \approx 1.725$ and showed a lower bound of 1.565, as $m \to \infty$, on the best possible competitive ratio of deterministic strategies. Cheng et al. [19] presented a deterministic 1.6-competitive semi-online algorithm and gave a lower bound of 1.5, for $m \geq 6$, on the competitiveness of deterministic strategies.

Our Contribution In this section we present two contributions for semi-online scheduling, complementing the existing results for classical makespan minimization. We give a very simple deterministic semi-online algorithm that is based on an approach different from that of previous strategies. The algorithms by Angelelli et al. [7] and Cheng et al. [19] both resort to job classes, i. e. incoming jobs are classified according to their processing times. The best known strategy by Cheng et al. [19] uses five job classes. The algorithm consists of sophisticated job packing schemes. Over the course of the algorithm and its analysis two scheduling phases with two associated stages and up to eight (or ten) machine types have to be considered.

Instead in this chapter we develop an algorithm that does not resort to job classes. Our strategy is 1.75-competitive and hence does not achieve the best possible competitiveness. However, as mentioned above, the algorithm is very simple, see Section 2.2. An incoming job is either scheduled on the least loaded machine or on the machine with the $\lceil m/2 \rceil$-th highest load. The decision which of the two machines to choose depends on the least loaded machine. The analysis of the algorithm relies on a potential function that keeps track of accumulated load on all the machines when the least loaded machine has a certain load. We remark that our scheduling algorithm is more in the spirit of online scheduling strategies not knowing p^+, which achieve a competitiveness around 1.92. Hence our upper bound also highlights the additional power of a small piece of advice when provided to an online algorithm. We show that our analysis is tight, i. e. our algorithm does not achieve a competitive ratio smaller than 1.75. Moreover, we observe that the algorithm can be extended easily to the scenario where an online scheduler knows the value of the optimum makespan.

Secondly, we develop a new lower bound on the competitive ratio that can be achieved by deterministic semi-online algorithms. We show that the competitiveness is at least $c \geq 1.58504$, as $m \to \infty$. This ratio almost matches the upper bound of 1.6 presented by Cheng et al. [19]. Formally, the lower bound c is the root of the function $f(x) = 4x^3 - 8x^2 + 2x + 1$ that is in the range $[1.58504, 1.58505]$. We note that c is greater than $e/(e-1)$, which is a ratio often appearing in the analysis of online algorithms. Our lower bound proof consists of an explicit construction of an adversarial job sequence. It does not rely on numerical techniques or computer assisted proofs.

2.2 A Semi-Online Algorithm without Job Classes

In this section we present a semi-online algorithm that is based on an approach different from that of previous strategies [7, 19] and does not rely on job classes. The algorithm is called LIGHT LOAD, or LL for short, because it tries to keep the

- Each job J_t is scheduled as follows
  - If $\ell(m,t) > 0.25\frac{p^+}{m}$ and $\ell(j_0,t) + p_t \leq 1.75\frac{p^+}{m}$, then assign J_t to M_{j_0}.
  - Otherwise assign J_t to M_m.

Figure 2.1: Algorithm LL.

least loaded machine, and in fact $\lfloor m/2 \rfloor$ machines, lightly loaded.

During the scheduling process the algorithm always maintains a list of the m machines sorted in non-increasing order of current load. Hence, for ease of presentation, we use the following slightly deviating notation in this section. At each time t we refer with M_j to the machine having the j-th highest load in LL's schedule at time t. In particular, M_1 is a machine with the highest load, i. e. $\ell(1,t) \geq \ell(j,t)$ for all $1 \leq j \leq m$, and M_m is a least loaded machine, i. e. $\ell(m,t) \leq \ell(j,t)$ for all $1 \leq j \leq m$. Notice that two loads $\ell(j,t)$ and $\ell(j,t')$, for $t \neq t'$, thus may refer to distinct machines. We require w. l. o. g. the ordering of machines by load to be stable in the following sense. If there held $\ell(j,t) \geq \ell(j',t)$, with $j < j'$, and the load of the machine $M_{j'}$ does still not exceed the load of machine M_j at time $t+1$, then the machine to which we referred as M_j at time t still has a smaller index than the machine to which we referred with $M_{j'}$ at time t. This means that the relative order of the machines in the sequence $M_1, \ldots, M_m$ remains unchanged when their loads do not change. Below let always $j_0 = \lceil m/2 \rceil$. Of specific interest at each time is machine M_{j_0} having the $\lceil m/2 \rceil$-th highest load.

Algorithm LL processes a job sequence $\sigma = J_1, \ldots, J_n$ as follows. When a new job J_t arrives, it is assigned to machine M_m if $\ell(m,t) \leq 0.25\frac{p^+}{m}$. When $\ell(m,t) > 0.25\frac{p^+}{m}$, then LL prefers to schedule J_t on machine M_{j_0}. The algorithm checks if such an assignment is possible without exceeding a load of $1.75\frac{p^+}{m}$. If indeed $\ell(j_0,t) + p_t \leq 1.75\frac{p^+}{m}$, then J_t is scheduled on M_{j_0}; otherwise J_t is assigned to the least loaded machine M_m. A summary of the algorithm is given in Figure 2.1.

We explain the choice of the algorithm's parameters. The proof that LL is 1.75-competitive crucially depends on the definition of $j_0 = \lceil m/2 \rceil$. We will show that if $\ell(m,t) > 0.75\frac{p^+}{m}$, and hence a new job J_t cannot necessarily be scheduled such that the resulting makespan is upper bounded by $1.75\frac{p^+}{m}$, then the job sequence contains $m+1$ large jobs of processing time greater than $0.5\frac{p^+}{m}$. In order to secure the existence of these jobs, we need to show that (a) M_{j_0} has a load of at most $1.25\frac{p^+}{m}$ and (b) M_m had a load of at most $0.25\frac{p^+}{m}$ over a long time horizon. The

load bounds of (a) and (b) only hold if $j_0 = \lceil m/2 \rceil$. In this case it is possible to balance load between $\lceil m/2 \rceil$ heavily loaded and $\lfloor m/2 \rfloor$ lightly loaded machines. Any other choice of j_0 will lead to a higher competitive ratio. Moreover, LL works with a load bound of $0.25\frac{p^+}{m}$ for machine M_m. This ensures that an assignment of a small job of processing time at most $0.5\frac{p^+}{m}$ does not exceed the critical load threshold of $0.75\frac{p^+}{m}$.

Theorem 2.1. *Algorithm* LL *achieves a competitive ratio of 1.75.*

In the following we prove the above theorem. We show that for fixed μ and any job sequence σ

$$\text{LL}(\mu, \sigma) \leq 1.75 \cdot \text{OPT}(\mu, \sigma). \tag{2.1}$$

The proof is by induction on the length n of the job sequence σ. We firstly show two lemmas, which give the above stated properties (a) and (b). Assume $\ell(m, n) > 0.75\frac{p^+}{m}$. In the following let $\delta > 0$ such that $\ell(m, n) = (0.75 + \delta)\frac{p^+}{m}$. Call a machine M_j full (at time t) if $\ell(j, t + 1) \geq (0.75 + \delta)\frac{p^+}{m}$. Throughout this subsection let t_j, for $1 \leq j \leq m$, denote the times when there were exactly j full machines in LL's schedule. As M_m is a least loaded machine, at time $n - 1$ all machines are full under the above assumption that $\ell(m, n) = (0.75+\delta)\frac{p^+}{m}$. Hence if $\ell(m, n) > 0.75\frac{p^+}{m}$, the times t_j, for $1 \leq j \leq m$, exist and $t_j < n$ for each j. The following lemma shows the above property (b).

Lemma 2.2. *Assume $\ell(m, n) > 0.75\frac{p^+}{m}$. Suppose a job J_n, with $p_n \geq (0.5+\delta)\frac{p^+}{m}$, needs to be assigned. There holds $\ell(m, t_{j_0} + 1) \leq 0.25\frac{p^+}{m}$.*

Proof. We will assume $\ell(m, t_{j_0} + 1) > 0.25\frac{p^+}{m}$ and derive a contradiction to the fact that the total processing time of jobs in σ is p^+. For the further analysis we need a potential function Φ whose definition is based on a machine set $\mu_f(t)$, for $t_{j_0} \leq t \leq t_{j_m}$. Let $\mu_f(t_{j_0}) := \{M_j \in \mu \mid \ell(j, t_{j_0} + 1) \geq (0.75 + \delta)\frac{p^+}{m}\}$ be the set of machines that are full at time t_{j_0}. At times $t > t_{j_0}$ we update this set whenever a machine becomes full. More precisely, for any time $t > t_{j_0}$, if $t \neq t_j$, for all $j = j_0 + 1, \ldots, m$, then $\mu_f(t) = \mu_f(t - 1)$. If $t = t_j$, for some j with $j_0 + 1 \leq j \leq m$, then $\mu_f(t_j)$ is obtained from $\mu_f(t_j - 1)$ by deleting the machine M_j having the smallest load $\ell(j, t_j)$ in $\mu_f(t_j - 1)$. In case of ties, the machine $M_j \in \mu_f(t_j - 1)$ with the highest index in the machine ordering $M_1, \ldots, M_m$ is chosen.

Since at any time t_j, for $j = j_0 + 1, \ldots, m$, exactly one machine is deleted from the set and $j_0 = \lceil m/2 \rceil \geq m - \lceil m/2 \rceil = m - j_0$, set $\mu_f(t)$ is non-empty at all times t with $t_{j_0} \leq t < t_m$, and hence $\mu_f(t)$ is well-defined for all $t_{j_0} \leq t \leq t_m$.

Define

$$\Phi(t) = \sum_{M_j \in \mu_f(t)} (\ell(j, t+1) - 0.75\tfrac{p^+}{m}).$$

Intuitively, Φ is the total load in excess to $0.75\frac{p^+}{m}$ on the machines of $\mu_f(t)$. Since every machine of $\mu_f(t_{j_0})$ has a load of at least $(0.75+\delta)\frac{p^+}{m}$ and machine loads can only increase, Φ is always non-negative.

We next argue that at all times t with $t_{j_0} \leq t < t_m$, all machines of $\mu_f(t)$ are among the j_0 machines having the highest load in LL's current schedule. More formally, at any time t, $t_{j_0} \leq t < t_m$, let $\mu_h(t)$ denote the set consisting of the j_0 machines $M_1, \ldots, M_{j_0}$ with highest load at time $t+1$, i. e. $\mu_h(t)$ contains the j_0 machines with highest values $\ell(j, t+1)$. We will show $\mu_f(t) \subseteq \mu_h(t)$. Recall that we assume w. l. o. g. that whenever machines are sorted according to their load after a scheduling step, only the rank of the machine that received the new job changes, and the relative order of all the other machines remains unchanged. In other words, machines having equal load appear in the same order before and after the scheduling step.

Obviously $\mu_f(t_{j_0}) \subseteq \mu_h(t_{j_0})$ is satisfied because at time t_{j_0} there exist exactly j_0 full machines in LL's schedule and $\mu_f(t_{j_0})$ contains all these machines. So suppose that $\mu_f(t-1) \subseteq \mu_h(t-1)$ holds, where $t_{j_0} < t \leq t_m - 1$, and consider the scheduling step at time t. Algorithm LL assigns the incoming job J_t either to machine M_{j_0} with the j_0-th highest load at time t or to machine M_m, the least loaded machine at time t. If J_t is placed on machine M_{j_0}, then $\mu_h(t-1) = \mu_h(t)$. If J_t is assigned to the least loaded machine M_m and the machine does not become full, then again $\mu_h(t-1) = \mu_h(t)$ because set $\mu_h(t)$ only contains full machines. Hence $\mu_h(t) \neq \mu_h(t-1)$ only if J_t is assigned to M_m and M_m becomes full.

If $\mu_h(t) = \mu_h(t-1)$, then $\mu_f(t) \subseteq \mu_f(t-1) \subseteq \mu_h(t-1) = \mu_h(t)$ and we are done. So assume that $\mu_h(t) \neq \mu_h(t-1)$. As argued in the last paragraph J_t is placed on the least loaded machine M_m and this machines becomes full. Thus $t = t_j$, for some j with $j_0 < j \leq m-1$. At this time the former machine M_m is added to $\mu_h(t_j)$ while the former machine $M_{j_0} \in \mu_h(t_j - 1)$ is removed from this set and is not in $\mu_h(t)$. Note that M_{j_0} is a least loaded machine in $\mu_h(t_j - 1)$. At time t_j, the least loaded machine in $\mu_f(t_j - 1)$ is removed from this set; in case of ties the highest indexed machine is chosen. Since both sets $\mu_f(t_j - 1)$ and $\mu_h(t_j - 1)$ lose least loaded machines, property $\mu_f(t_j - 1) \subseteq \mu_h(t_j - 1)$ implies $\mu_f(t_j) \subseteq \mu_h(t_j)$.

For any t let $p_t^+ = \sum_{i=1}^{t} p_i$ denote the total processing time of the jobs $J_1, \ldots, J_t$. We will show that if $\ell(m, t_{j_0}+1) > 0.25\frac{p^+}{m}$, then the following inequality holds for $j = j_0, \ldots, m$.

$$p_{t_j}^+ - \Phi(t_j) > 0.25p^+ + j_0 \cdot 0.5\tfrac{p^+}{m} + (j - j_0)\tfrac{p^+}{m} \tag{2.2}$$

Using (2.2) for $j = m$ and observing again that the potential is non-negative, we obtain that at time $t_m + 1$ and hence before the assignment of J_n, the total processing time of jobs scheduled so far is at least

$$\begin{aligned} p^+_{t_m} &> 0.25p^+ + \lceil \tfrac{m}{2} \rceil 0.5\tfrac{p^+}{m} + (m - \lceil \tfrac{m}{2} \rceil)\tfrac{p^+}{m} \\ &\geq 1.25p^+ - (\tfrac{m}{2} + \tfrac{1}{2})0.5\tfrac{p^+}{m} \\ &= p^+ - 0.25\tfrac{p^+}{m}. \end{aligned}$$

Since J_n has a processing time of $p_n > 0.5\frac{p^+}{m}$, the total processing time of jobs in σ is at least $p^+_{t_m} + p_n > p^+ + 0.25\frac{p^+}{m} > p^+$. We obtain the desired contradiction.

It remains to show (2.2), for $j = j_0, \ldots, m$, assuming that $\ell(m, t_{j_0} + 1) > 0.25\frac{p^+}{m}$ holds. The proof is by induction on j. First consider j_0. At time t_{j_0} exactly j_0 machines are full and thus $\ell(j, t_{j_0} + 1) > 0.75\frac{p^+}{m}$ for $1 \leq j \leq j_0$. By assumption $\ell(m, t + 1) > 0.25\frac{p^+}{m}$. Hence machines $M_{j_0+1}, \ldots, M_m$ each have a load greater than $0.25\frac{p^+}{m}$ and the total load on these $m - j_0$ machines is greater than $(m - j_0) \cdot 0.25\frac{p^+}{m}$. We obtain that the total load on the m machines is

$$\begin{aligned} \sum_{j=1}^{m} \ell(j, t_0 + 1) &= \sum_{j=1}^{j_0} \ell(j, t_{j_0} + 1) + \sum_{j=j_0+1}^{m} \ell(j, t_{j_0} + 1) \\ &= j_0 \cdot 0.75\tfrac{p^+}{m} + \sum_{j=1}^{j_0} (\ell(j, t_{j_0} + 1) - 0.75\tfrac{p^+}{m}) \\ &\quad + \sum_{j=j_0+1}^{m} \ell(j, t_{j_0} + 1) \\ &> j_0 \cdot 0.75\tfrac{p^+}{m} + \sum_{j=1}^{j_0} (\ell(j, t_{j_0} + 1) - 0.75\tfrac{p^+}{m}) + (m - j_0)0.25\tfrac{p^+}{m} \\ &= 0.25\tfrac{p^+}{m} + j_0 \cdot 0.5\tfrac{p^+}{m} + \Phi(t_{j_0}). \end{aligned}$$

The last equation holds because machines $M_1, \ldots, M_{j_0}$ form set $\mu_f(t_{j_0})$. Inequality (2.2) then follows for $j = j_0$ since $p^+_{t_0} = \sum_{j=1}^{m} \ell(j, t_0 + 1)$.

Next suppose that (2.2) holds for index j. We show that it is also satisfied for $j + 1$. We first argue that

$$p^+_t - \Phi(t) > 0.25p^+ + j_0 \cdot 0.5\tfrac{p^+}{m} + (j - j_0)\tfrac{p^+}{m} \tag{2.3}$$

holds for any $t = t_j, t_j + 1, \ldots, t_{j+1} - 1$. By induction hypothesis the above inequality holds for $t = t_j$. By choice of the times t_j and t_{j+1} there holds $\mu_f(t) =$

$\mu_f(t_j)$ for $t_j < t < t_{j+1}$. At any time t the incoming job J_t increases the total load on the m machines by p_t, i. e. $p^+_{t-1} + p_t = \sum_{j=1}^{m} \ell(j,t) + p_t = \sum_{j=1}^{m} \ell(j,t+1) = p^+_t$. The potential Φ only increases by p_t if J_t is assigned to a machine in $\mu_f(t_j)$. Hence the left hand side of (2.3) does not decrease at times $t = t_j+1, \ldots, t_{j+1}-1$.

At time t_{j+1} another machine becomes full. An assignment to a machine M_{j_0} at time $t > t_j$ would not generate an additional full machine. This holds true because at time t_0 there already exist j_0 machines that are full and hence also at time $t_{j+1} > t_{j_0}$ since loads do not decrease. Thus a machine can only become full if the incoming job $J_{t_{j+1}}$ is placed on the least loaded machine M_m at time $t_{j+1} = t$. By assumption $\ell(m, t_{j_0}+1) > 0.25\frac{p^+}{m}$ and hence also $\ell(m, t_{j_1}) > 0.25\frac{p^+}{m}$. Thus LL would prefer to schedule $J_{t_{j+1}}$ on machine M_{j_0}. Since this assignment is not performed, the resulting load would exceed $1.75\frac{p^+}{m}$, i. e. $\ell(j_0, t_{j+1}) + p_{t_{j+1}} > 1.75\frac{p^+}{m}$ and hence

$$p_{t_{j+1}} > \tfrac{p^+}{m} - (\ell(j_0, t_{j+1}) - 0.75\tfrac{p^+}{m}).$$

Machine M_{j_0} is a least loaded machine in $\mu_h(t_{j+1}-1)$. At time t_{j+1} the least loaded machine in $\mu_f(t_{j+1}-1)$ is removed from the set. As argued above $\mu_f(t) \subseteq \mu_h(t)$ for any t with $t_{j_0} \le t < t_m$. Hence the least loaded machine in $\mu_f(t_{j+1}-1)$ has a load of at least $\ell(j_0, t_{j+1})$. Thus at time t_{j+1} the potential decreases by at least

$$\Phi(t_{j+1}-1) - \Phi(t_{j+1}) \ge \ell(j_0, t_{j+1}) - 0.75\tfrac{p^+}{m}.$$

We obtain $p_{t_{j+1}} + \Phi(t_{j+1}-1) - \Phi(t_{j+1}) > \frac{p^+}{m}$ and, as desired,

$$\begin{aligned} p^+_{t_{j+1}} - \Phi(t_{j+1}) &= p^+_{t_{j+1}-1} - \Phi(t_{j+1}-1) + p_{t_{j+1}} + \Phi(t_{j+1}-1) - \Phi(t_{j+1}) \\ &> 0.25p^+ + j_0 \cdot 0.5\tfrac{p^+}{m} + (j+1-j_0)\tfrac{p^+}{m}. \end{aligned}$$

The inductive step is complete. □

Lemma 2.3. *Assume $\ell(m,n) > 0.75\frac{p^+}{m}$. Suppose a job J_n, with $p_n \ge (0.5+\delta)\frac{p^+}{m}$, needs to be assigned. At any time t, with $t_{j_0} \le t \le t_m$, there holds for machine M_{j_0} in* LL*'s schedule that $\ell(j_0, t) \le (1.25-\delta)\frac{p^+}{m}$.*

Proof. Suppose that at some time t, $t_{j_0} \le t \le t_m$, the machine M_{j_0} with the j_0-th highest load had a load greater than $(1.25-\delta)\frac{p^+}{m}$. Thus at this time t and also at time t_m+1 the j_0 machines with highest load in LL's schedule had a total load greater than $j_0(1.25-\delta)\frac{p^+}{m}$. At time t_m all machines of LL are full and hence at time t_m+1 the $m-j_0$ machines with the smallest load have a total load of at least

$(m - j_0)(0.75 + \delta)\frac{p^+}{m}$. Thus at time $t_m + 1$ the total load on the m machines is greater than

$$j_0(1.25-\delta)\tfrac{p^+}{m}+(m-j_0)(0.75+\delta)\tfrac{p^+}{m} \geq \lceil \tfrac{m}{2} \rceil 0.5\tfrac{p^+}{m}+0.75p^+-\delta\tfrac{p^+}{m} \geq p^+-\delta\tfrac{p^+}{m}.$$

Including J_n, whose load is not taken into account by values $\ell(j, t_m + 1)$ because of $n > t_m$, the total processing time of jobs in σ is greater than $p^+-\delta\frac{p^+}{m}+p_n > p^+$ because $p_n \geq (0.5 + \delta)\frac{p^+}{m}$. This contradicts the fact that the total processing time of jobs in σ is equal to p^+. □

We now can show Theorem 2.1.

Proof. (*of Theorem 2.1).* For job sequences consisting of a single job J_1 there is nothing to show because LL and OPT both have a makespan equal to the processing time p_1 of J_1. Suppose that (2.1) holds for job sequences of up to $n - 1$ jobs. We will prove that (2.1) is also satisfied for sequences consisting of n jobs.

Let $\sigma = J_1, \ldots, J_n$ be an arbitrary job sequence of length n. By induction hypothesis LL schedules the first $n - 1$ jobs such that a performance ratio of 1.75 is maintained, i. e. LL assigns each job such that its resulting makespan is at most 1.75 times the optimum makespan for the job sequence processed so far. In the following we investigate the assignment of J_n and prove that the scheduling step also maintains the desired performance guarantee. We concentrate on the case that the assignment of J_n causes an increase in LL's makespan since otherwise there is nothing to show.

If LL schedules J_n on machine M_{j_0}, we are easily done because by the definition of the algorithm $\ell(j_0, n) + p_n \leq 1.75\frac{p^+}{m}$. The ratio $\frac{p^+}{m}$ is a lower bound on the optimum makespan and hence $\ell(j_0, n) + p_n \leq 1.75 \cdot \text{OPT}(\mu, \sigma)$. Moreover, if LL schedules J_n on the least loaded machine M_m and $\ell(m, n) \leq 0.75\frac{p^+}{m}$, the analysis is simple: If $p_n \leq \frac{p^+}{m}$, then LL's resulting makespan is $\ell(m, n + 1) = \ell(m, n) + p_n \leq 0.75\frac{p^+}{m} + \frac{p^+}{m} \leq 1.75\frac{p^+}{m} \leq 1.75 \cdot \text{OPT}(\mu, \sigma)$. If $p_n > \frac{p^+}{m}$, then $\ell(m, n) + p_n \leq 0.75p_n + p_n = 1.75p_n \leq 1.75 \cdot \text{OPT}(\mu, \sigma)$ because the optimum makespan on (μ, σ) cannot be smaller than the processing time of any job.

Therefore we can restrict ourselves to the case that LL schedules J_n on M_m and $\ell(m, n) > 0.75\frac{p^+}{m}$. Let $\ell(m, n) = (0.75 + \delta)\frac{p^+}{m}$, for some $\delta > 0$. We have $\delta < 0.25$ because M_m is a least loaded machine and hence its load $\ell(m, n)$ is smaller than $\frac{p^+}{m}$. If we had $\ell(m, n) \geq \frac{p^+}{m}$, then all machines would have a load of at least $\frac{p^+}{m}$ and the total load on the m machines at the arrival of J_n would be $m \cdot \frac{p^+}{m} = p^+$. Hence the total processing time of jobs in σ would be at least $p^+ + p_n > p^+$, contradicting the fact that total processing volume equals p^+. Thus $0 < \delta < 0.25$. If $\ell(m, n) + p_n \leq 1.75\frac{p^+}{m}$, we are again done. Hence we

assume $\ell(m,n) + p_n > 1.75\frac{p^+}{m}$. We obtain $p_n > 0.75\frac{p^+}{m}$ because, as just argued, $\ell(m,n) < \frac{p^+}{m}$. Therefore, $p_n > (0.5+\delta)\frac{p^+}{m}$.

In the following we will show that at time n, each machine in LL's schedule contains a job of processing time at least $(0.5+\delta)\frac{p^+}{m}$. This implies that, including J_n, the job sequence σ contains $m+1$ jobs of processing time at least $(0.5+\delta)\frac{p^+}{m}$ each. Two of these jobs must be scheduled on the same machine in an optimal schedule and hence $\text{OPT}(\mu,\sigma) \geq (1+2\delta)\frac{p^+}{m}$. Using this property we can prove the theorem. If $p_n \leq (1+2\delta)\frac{p^+}{m}$, then LL's resulting makespan is $\ell(m,n+1) = \ell(m,n) + p_n \leq (0.75+\delta)\frac{p^+}{m} + (1+2\delta)\frac{p^+}{m} = (1.75+3\delta)\frac{p^+}{m} \leq 1.75(1+2\delta)\frac{p^+}{m} \leq 1.75 \cdot \text{OPT}(\mu,\sigma)$. If $p_n > (1+2\delta)\frac{p^+}{m}$, then the resulting makespan is $\ell(m,n+1) = \ell(m,n)+p_n \leq (0.75+\delta)\frac{p^+}{m}+p_n \leq (0.75+\delta)p_n/(1+2\delta)+p_n = (1.75+3\delta)p_n/(1+2\delta) \leq 1.75 \cdot p_n \leq 1.75 \cdot \text{OPT}(\mu,\sigma)$.

It remains to prove that immediately before the assignment of J_n each machine in LL's schedule contains a job of processing time at least $(0.5+\delta)\frac{p^+}{m}$. When J_n arrives, the least loaded machine and hence any machine in LL's schedule is full since $\ell(m,n) = (0.75+\delta)\frac{p^+}{m}$. We consider the past scheduling steps of the jobs $J_1,\ldots,J_{n-1}$. Recall t_j, $1 \leq j \leq m$, is the first point of time when exactly j machines are full in LL's schedule, i. e. the assignment of J_{t_j} causes the j-th machine to become full. We have $1 \leq t_1 < \ldots < t_m \leq n-1$. Of particular interest is the time t_{j_0} when exactly j_0 machines are full.

We show that at any time t_j, for $j = 1,\ldots,m$, a job of processing time at least $(0.5+\delta)\frac{p^+}{m}$ is scheduled. By Lemma 2.2 we have $\ell(m,t_{j_0}+1) \leq 0.25\frac{p^+}{m}$. Hence at any time with $t \leq t_{j_0}$ we have $\ell(m,t) \leq 0.25\frac{p^+}{m}$ and LL schedules an incoming job on the least loaded machine. At any time t_j with $j = 1,\ldots,j_0$ a machine becomes full and hence $\ell(m,t_j) + p_{t_j} \geq (0.75+\delta)\frac{p^+}{m}$. This implies $p_{t_j} \geq (0.75+\delta)\frac{p^+}{m} - 0.25\frac{p^+}{m} = (0.5+\delta)\frac{p^+}{m}$.

Next consider the times t_j with $j_0 < j \leq m$. At those times another full machine can only be created if the incoming job is scheduled on the least loaded machine. Let t^* be the first point of time at which the least loaded machine in LL's schedule has a load greater than $0.25\frac{p^+}{m}$. As above we can show that at any time t_j with $t_{j_0} < t_j \leq t^*$ a job of processing time at least $(0.5+\delta)\frac{p^+}{m}$ is scheduled. Finally consider times t_j with $t^* < t_j \leq t_m$. The least loaded machine in LL's schedule has a load greater than $0.25\frac{p^+}{m}$. Therefore, LL would prefer to place J_{t_j} on the machine with the j_0-th highest load. Since J_{t_j} is placed on the least loaded machine instead, we have $\ell(j_0,t_j) + p_{t_j} > 1.75\frac{p^+}{m}$. By Lemma 2.3, $\ell(j_0,t_j) \leq (1.25-\delta)\frac{p^+}{m}$ and hence $p_{t_j} > (0.5+\delta)\frac{p^+}{m}$. This concludes the proof of Theorem 2.1. □

We next provide a matching lower bound on the performance of LL.

Theorem 2.4. *Algorithm* LL *does not achieve a competitive ratio smaller than 1.75.*

Proof. For simplicity we assume that m is even. Moreover, let $m \geq 4$. Choose an ε with $0 < \varepsilon < 1$. We prove that LL does not achieve a competitive ratio smaller than $1.75-\varepsilon$. Let $p^+ = m$. Furthermore, let k be an integer satisfying $k \geq 1.75/\varepsilon$ and set $s_1 = 1/(4k)$. An adversary first presents km jobs of processing time s_1. These jobs have a total processing time of $m/4$. Thus, while the jobs of processing time s_1 arrive, machine M_m in LL's schedule has a load of at most $1/4 = 0.25\frac{p^+}{m}$ and each job with processing time s_1 is assigned to this least loaded machine M_m. Hence, when all the km jobs of processing time s_1 are scheduled, each of the m machines has a load of exactly $ks_1 = 0.25$.

Next the adversary presents $m/2$ jobs of processing time $s_2 = 0.5$ and $m/2$ jobs of processing time $s_3 = 1 - 2/m$. While these jobs are scheduled, the least loaded machine M_m in LL's schedule has a load of exactly $0.25\frac{p^+}{m}$. Thus, again, any of these m jobs is placed on machine M_m. After the assignment of these jobs, each machine in LL's schedule has a load of at least 0.75 because $s_3 = 1 - 2/m \geq 0.5$. The adversary finally reveals a job of processing time $s_4 = 1$ so that LL's final makespan is 1.75.

The total processing time of all jobs is $kms_1 + m/2 \cdot s_2 + m/2 \cdot s_3 + 1 = m/4 + m/4 + m/2(1 - 2/m) + 1 = m = p^+$, as desired. The adversary can construct a schedule whose makespan is upper bounded by $1 + s_1$: The $m/2$ jobs of processing time s_3 and the job of processing time s_4 are assigned to distinct machines. The $m/2$ jobs of processing time $s_2 = 0.5$ are combined to pairs and placed on machines on which no jobs with processing time s_3 or s_4 reside. If $m/2$ is odd, then one job with processing time s_2 is scheduled alone on a machine. Finally, each job of processing time s_1 is scheduled on a machine currently having the smallest load.

We conclude that the competitive ratio of LL is at least $1.75/(1 + s_1)$ and this ratio is at least $1.75 - \varepsilon$ because $s_1 < \varepsilon/1.75$. □

We finally observe that LL can be extended easily to the scenario where an online scheduler knows the value $\text{OPT}(\mu, \sigma)$ of the optimum makespan. In this case we just have to replace $\frac{p^+}{m}$ by $\text{OPT}(\mu, \sigma)$ in both the description and the analysis of the algorithm.

Corollary 2.5. *Algorithm* LL *achieves a competitive ratio of 1.75 if p^+/m is replaced by the value of the optimum makespan* $\text{OPT}(\mu, \sigma)$.

2.3 A New Lower Bound

In this section we present a lower bound on the competitive ratio that can be achieved by deterministic semi-online algorithms. Consider the function $f(x) = 4x^3 - 8x^2 + 2x + 1$. This function has three real-valued roots, one of which is in the range $[1.58504, 1.58505]$. The lower bound is equal to this root. The other two roots of f are in the ranges $[-0.25, -0.24]$ and $[0.65, 0.66]$.

Theorem 2.6. *Let c denote the root of $f(x) = 4x^3 - 8x^2 + 2x + 1$, with $c \in [1.58504, 1.58505]$. No deterministic semi-online algorithm can be ρ-competitive, for $\rho < c$, as $m \to \infty$.*

Proof. Let ALG be any deterministic semi-online algorithm. In the following c always denotes the value as specified in the statement of the theorem. The adversary presents a job sequence σ in which the total processing time of the jobs is equal to $p^+ = m + 16c^2 - 12c - 16$. We remark that the expression $16c^2 - 12c - 16$ is upper bounded by 5.2. The exact structure of σ depends on the behavior of ALG, but in each case the adversary uses at most four different processing times, which we denote by s_i, $1 \leq i \leq 4$. The following construction of σ is possible for any $m > 8$.

Initially, the adversary presents $m - 4$ jobs of processing time $s_1 = 1$ each. If ALG assigns two of these jobs to the same machine, then the adversary presents four additional jobs with a processing time of $s_1 = 1$ succeeded by m jobs with a processing time of $s_2 = (16c^2 - 12c - 16)/m$. Algorithm ALG has a makespan of at least 2 while the adversary has a makespan of $1 + s_2$ only. In this case the ratio of ALG's makespan to the adversary's makespan can be arbitrarily close to 2, as $m \to \infty$.

In the following we concentrate on the case in which ALG places the $m - 4$ jobs with processing time s_1 on different machines. At this point the schedule of ALG has four empty machines. The adversary presents four jobs of processing time $s_2 = c - 1$. We distinguish three cases.

(1) Algorithm ALG assigns one job with processing time s_2 to a machine already containing a job of processing time s_1.

(2) Algorithm ALG assigns the jobs with processing time s_2 only to machines that do not contain a job of processing time s_1 yet, and two jobs with processing time s_2 are placed on the same machine.

(3) Algorithm ALG assigns all jobs with processing time s_2 to different machines, and none of these already contains a job of processing time s_1.

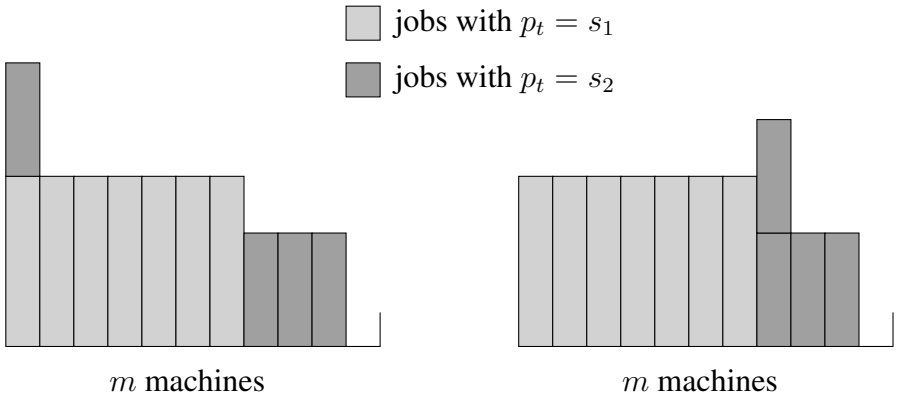

Figure 2.2: Case (1)

Figure 2.3: Case (2)

Figures 2.2 and 2.3 depict ALG's schedules in Cases (1) and (2), respectively. We next analyze the various cases.

Case (1): When the jobs of processing time s_2 are scheduled, ALG has a makespan of $s_1 + s_2 = 1 + c - 1 = c$ because there is a machine containing a job with processing time s_1 as well as a job with processing time s_2. The adversary completes the request sequence by presenting four jobs of processing time $s_3 = 2 - c$ and m jobs of processing time $s_4 = (16c^2 - 12c - 16)/m$. The sum of the jobs' processing times is $p^+ = (m-4)\cdot 1 + 4(c-1) + 4(2-c) + m(16c^2 - 12c - 16)/m = m + 16c^2 - 12c - 16$, as desired. The adversary constructs a schedule in which the jobs of processing time s_1 are assigned to different machines. Each job with processing time s_2 is paired with a job of processing time s_3 yielding a total processing time of $c - 1 + 2 - c = 1$. Each such a job pair is assigned to an empty machine. Finally, each of the m machines receives a job of processing time s_4. Thus the optimum makespan is no larger than $1 + s_4$. The ratio of ALG's makespan to the optimum makespan is hence at least $c/(1 + s_4)$ and this ratio tends to c as $m \to \infty$.

Case (2): As ALG has combined two jobs of processing time s_2, there is a machine in the schedule of ALG having a load of at least $2s_2 = 2(c - 1) > 1$. There are $m - 4$ additional machines containing a job of processing time s_1 and thus having a load of 1. Hence, when the jobs of processing time s_2 are scheduled, there exist at most three machines having a load smaller than 1. The adversary next reveals four jobs each with a processing time of $s_3 = c$. Algorithm ALG must place at least one of them on a machine with a load at least 1, incurring a makespan of at least $1+c$. The adversary completes the request sequence by presenting $m-8$ jobs of processing time $s_4 = (16c^2-20c-8)/(m-8)$. The sum of the processing times of the jobs is $p^+ = (m-4)\cdot 1 + 4(c-1) + 4c + (m-8)(16c^2-20c-8)/(m-8) =$

$m + 16c^2 - 12c - 16$, as claimed. There exists a schedule with makespan no more than c. The $m-4$ jobs of processing time s_1 are placed on distinct machines. Four of these machines receive an additional job with processing time s_2 and $m - 8$ of these are assigned an extra job of processing time s_4. The four remaining jobs of processing time s_3 are placed separately on the four left empty machines. We have $s_1 + s_2 = c$ and $s_1 + s_4 < c$ because $s_4 < 0.5$, for $m > 8$. Thus no machine has a load greater than c. We conclude that the ratio of ALG's makespan to the adversary's makespan is at least $(1 + c)/c$ and this expression is greater than c, for our choice of c.

Case (3): Algorithm ALG assigns the $m - 4$ jobs of processing time s_1 and the four jobs of processing time s_2 to different machines so that, after the assignment, each machine contains exactly one job and there is no empty machine in the schedule. The adversary presents two jobs of processing time $s_3 = 2c(c - 1) - 1$. Again we distinguish two cases.

(a) Algorithm ALG assigns a job of processing time s_3 to a machine containing a job with processing time s_1, or it assigns both jobs of processing time s_3 to the same machine containing a job of processing time s_2.

(b) Algorithm ALG assigns each job of processing time s_3 to a machine already containing a job with processing time s_2.

Figure 2.4 depicts ALG's schedule in Case (3a) if a job with processing time s_3 is assigned to a machine containing a job of processing time s_1. Figure 2.5 shows the schedule in Case (3b).

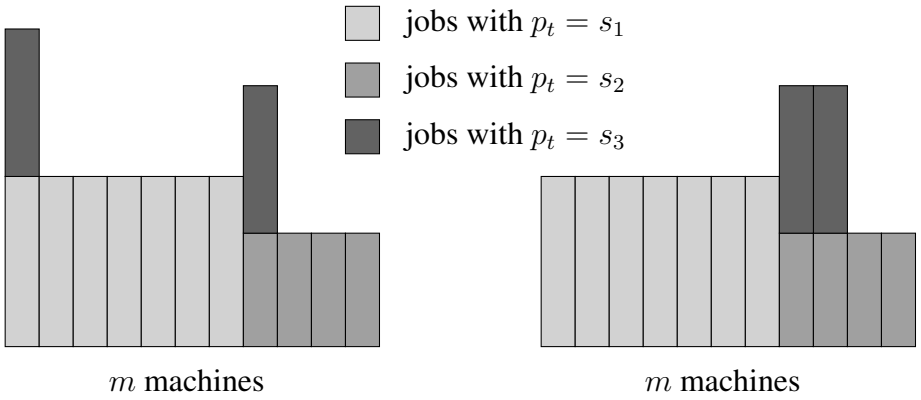

Figure 2.4: Case (3a)

Figure 2.5: Case (3b)

Case (3a): If a job of processing time s_3 is assigned to a machine containing a job of processing time s_1, then ALG has a makespan of at least $s_1 + s_3 = 2c(c-1)$.

In case that both jobs with processing time s_3 are placed on the same machine already containing a job with processing time s_2, the makespan of ALG is at least $s_2+2s_3 = c-1+2(2c(c-1)-1) = 2c(c-1)+(2c+1)(c-1)-2 > 2c(c-1)$ because $c > 1.5$. The adversary finishes the request sequence by sending $m-4$ jobs of processing time $s_4 = 6(2c(c-1)-1)/(m-4)$. The total processing time of the jobs is $p^+ = (m-4)s_1 + 4s_2 + 2s_3 + (m-4)s_4 = m-4+4(c-1)+2(2c(c-1)-1)+6(2c(c-1)-1) = m+16c^2-12c-16$.

The adversary constructs the following schedule. Each job having processing time s_1 is paired with a job having processing time s_4, and $m-4$ many machines receive such a pair. Two of the remaining four machines receive two jobs with processing time s_2 each. Finally, the two jobs of processing time s_3 are each assigned to one of the remaining empty machines. We have $2s_2 > s_3$ because this inequality is equivalent to $1 > 2(c-1)^2$ and is satisfied since $c < 1.6$. Moreover, for $m \geq 35$, we have $2s_2 > s_1 + s_4$ and the adversary's makespan is upper bounded by $2s_2 = 2(c-1)$. In summary, for $m \geq 35$ and hence for $m \to \infty$, the ratio of ALG's makespan to the adversary's makespan is $2c(c-1)/(2(c-1)) = c$.

Case (3b): Algorithm ALG assigns the two jobs of processing time s_3 each to a machine containing a job with processing time s_2. The load of these machines is $s_2+s_3 = c-1+2c(c-1)-1 = (2c+1)(c-1)-1 > 1$. Thus after the jobs of processing time s_3 are scheduled, there are only two machines in the schedule of ALG left that have a load smaller than 1. The adversary then presents three final jobs with a processing time $s_4 = 2s_3 = 2(2c(c-1)-1)$. Again, we have a total processing time of $p^+ = (m-4)s_1+4s_2+2s_3+3s_4 = m-4+4(c-1)+8(2c(c-1)-1) = m+16c^2-12c-16$. Algorithm ALG must schedule one of the jobs with processing time s_4 on a machine having a load of at least 1. Hence its makespan is at least $1+s_4 = 1+2s_3$. By contrast, the adversary can construct a schedule with a makespan of $s_4 = 2s_3$. The $m-4$ jobs with processing time s_1 are assigned to distinct machines. Four of these machines receive an additional job of processing time s_2, which results in a load of $1+s_2 = c < s_4$. One of the remaining four machines is assigned the two jobs with processing time s_3. The other three machines each receive a job with processing time s_4. Hence the ratio of ALG's makespan to the adversary's makespan is $(1+s_4)/s_4 = 1+1/s_4 = 1+1/(2s_3)$. We have $2(c-1)s_3-1 = 0$ because $2(c-1)s_3-1 = 4c(c-1)^2-2(c-1)-1 = 4c^3-8c^2+2c+1$ and c is a root of the function $f(x) = 4x^3-8x^2+2x+1$. Hence $2s_3 = 1/(c-1)$ and we conclude that the ratio of ALG's makespan to the adversary's makespan is $1+1/(2s_3) = 1+(c-1) = c$. □

Chapter 3

The Value of Job Migration in Minimum Makespan Scheduling

In this chapter we investigate the impact of job migration in ONLINE MINIMUM MAKESPAN SCHEDULING.

3.1 Introduction

The Model As before we are given a set $\mu = \{M_1, \ldots, M_m\}$ of m machines and a sequence $\sigma = J_1, \ldots, J_n$ of n jobs that arrive one by one. Recall that at time t job J_t is presented and has to be assigned to some machine from μ without any knowledge about jobs $J_{t'}$ with $t' > t$. In this chapter an algorithm may perform *reassignments* at any time t, i. e. a job already scheduled on a machine may be removed and transferred to another machine.

Previous Work Makespan minimization with job migration was first addressed by Aggarwal et al. [1]. They consider an offline setting. An algorithm is given a schedule, in which all jobs are already assigned, and a budget. The algorithm may perform job migrations up to the given budget. The authors design strategies that perform well with respect to the best possible solution that can be constructed with the budget. Online makespan minimization on $m = 2$ machines was considered in [47, 55]. The best competitiveness is 4/3. Sanders et al. [49] study an online setting in which before the assignment of each job J_t, jobs up to a total processing volume of βp_t may be migrated, for some constant β. For $\beta = 4/3$, they present a 1.5-competitive algorithm. They also show a $(1 + \varepsilon)$-competitive algorithm, for any $\varepsilon > 0$, where β depends exponentially on $1/\varepsilon$. The algorithms are robust in that the stated competitive ratios hold after each job assignment. However in this framework, over time, $\Omega(n)$ migrations may be performed and jobs of total processing volume $\beta \sum_{t=1}^{n} p_t$ may be moved.

Englert et al. [26] study online makespan minimization if an algorithm is given a buffer that may be used to partially reorder the job sequence. In each step an algorithm assigns one job from the buffer to the machines. Then the next job in σ is admitted to the buffer. Englert et al. show that, using a buffer of size $\Theta(m)$, the best competitive ratio is $W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2))$, where W_{-1} is the lower branch of the Lambert W function.

Our Contribution We investigate online makespan minimization with limited migration. Our work shows that even with a very limited number of migration operations, significantly improved performance guarantees are obtained. We develop tight upper and lower bounds on the competitive ratio that can be achieved by deterministic online algorithms. The number of job reassignments does not depend on the length of the job sequence. We determine the exact competitiveness achieved by deterministic algorithms, for general m.

In Section 3.2 we develop an optimal algorithm. For any $m \geq 2$, the strategy is ρ_m-competitive, where ρ_m is the solution of an equation representing load in an ideal machine profile for a subset of the jobs. For $m = 2$, the competitive ratio is 4/3. The ratios are non-decreasing and converge to $W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2)) \approx 1.4659$ as m tends to infinity. Again, W_{-1} is the lower branch of the Lambert W function. The algorithm uses at most $(\lceil (2 - \rho_m)/(\rho_m - 1)^2 \rceil + 4)m$ job migrations. For $m \geq 11$, this expression is at most $7m$. For smaller machine numbers it is $8m$ to $10m$. We note that the competitiveness of 1.4659 is considerably below the factor of roughly 1.9 obtained by deterministic algorithms in the standard online setting. It is also below the ratio of $e/(e - 1)$ attainable if randomization is allowed.

In Section 3.3 we give matching lower bounds. We show that no deterministic algorithm that uses $o(n)$ job migrations can achieve a competitive ratio smaller than ρ_m, for any $m \geq 2$. Hence in order to beat the factor of ρ_m, $\Theta(n)$ reassignments are required. Additionally, we show that $\Omega(m)$ migration is required to achieve a competitive ratio of $\rho < 1 + 1/\sqrt{2} \approx 1.707$, and hence our algorithm uses almost as few migration as possible to achieve the desired bound.

Finally, in Section 3.4 we trade migrations for performance. We develop a family of algorithms that is $\hat{\rho}$-competitive, for any constant $\hat{\rho}$ with $5/3 \leq \hat{\rho} \leq 2$. Setting $\hat{\rho} = 5/3$ we obtain a strategy that uses at most $4m$ job migrations. For $\hat{\rho} = 1.75$, the strategy uses no more than $2.5m$ migrations. Again, our algorithms use an almost optimal number of job migrations as the lower bound in Section 3.3 states that a migration of $\Omega(m)$ is really necessary to attain a competitiveness of $\hat{\rho} < 1 + 1/\sqrt{2}$.

Our algorithms rely on a number of ideas. All strategies classify incoming jobs into small and large depending on a careful estimate on the optimum makespan.

The algorithms consist of a job arrival phase followed by a migration phase. The optimal algorithm, in the arrival phase, maintains a load profile on the machines with respect to jobs that are currently small. In the migration phase, the algorithm removes a certain number of jobs from each machine. These jobs are then rescheduled using strategies by Graham [34, 35]. Our family of algorithms partitions the m machines into two sets μ_A and μ_B. In the arrival phase the algorithms prefer to place jobs on machines in μ_A so that machines in μ_B are available for later migration. In general, the main challenge in the analyses of the various algorithms is to bound the number of jobs that have to be migrated from each machine.

Relation to Previous Work We relate our contributions to some existing results. First we point out that the goal in online makespan minimization is to construct a good schedule when jobs arrive one by one. Once the schedule is constructed, the processing of the jobs may start. It is not stipulated that machines start executing jobs while other jobs of σ still need to be scheduled. This framework is assumed in all the literature on online makespan minimization mentioned above. Consequently it is no drawback to perform job migrations when the entire job sequence has arrived. Nonetheless, as for the algorithms presented in this chapter, the machines can start processing jobs except for the up to 10 largest jobs on each machine. A second remark is that the algorithms by Aggarwal et al. [1] cannot be used to achieve good results in the online setting. The reason is that those strategies are designed to perform well relative to the best possible makespan attainable from an initial schedule using a given migration budget. The strategies need not perform well compared to a globally optimal schedule. The algorithms by Aggarwal et al. and ours are different, see [1].

On the other hand, our results exhibit similarities to those by Englert et al. [26] where a reordering buffer is given. The optimal competitive ratio of ρ_m is the solution of an equation that also arises in [26]. This is due to the fact that our optimal algorithm and that in [26] maintain a certain load profile on the machines. Our strategy does so w. r. t. jobs that are currently small while the strategy in [26] considers all jobs assigned to machines. In our framework the profile is harder to maintain because of *shrinking jobs*, i. e. jobs that are large at some time t but small at later times $t' > t$. In the job migration phase our algorithm reschedules jobs removed from some machines. This operation corresponds to the "final phase" of the algorithm in [26]. However, our algorithm directly applies policies by Graham [34, 35] while the algorithm in [26] computes a virtual schedule.

In general, an interesting question is if makespan minimization with limited migration is equivalent to makespan minimization with a bounded reordering buffer. We cannot prove this in the affirmative. As for the specific algorithms presented in [26] and in this chapter, the following relation holds. All our algorithms

can be transformed into strategies with a reordering buffer. The competitive ratios are preserved and the number of job migrations is equal to the buffer size. This transformation is possible because our algorithms are *monotone*: If a job does not have to be migrated at time t, assuming σ ended at time t, then there is no need to migrate it at times $t' > t$. Hence, at any time a buffer can store the candidate jobs to be migrated. On the other hand, to the best of our knowledge, the algorithms by Englert et al. [26] do not translate into strategies with job migration. All the algorithms in [26] use the given buffer of size cm, for some constant c, to store the cm largest jobs of the job sequence. However in our setting, a migration of the largest jobs does not generate good schedules. The problem are shrinking jobs, i. e. jobs that are among the largest jobs at some time t but not at later times. We cannot afford to migrate all shrinking jobs, unless we invest $\Theta(n)$ migrations. With limited job migration, scheduling decisions are final for almost all of the jobs. Hence the corresponding algorithms are more involved than in the setting with a reordering buffer.

3.2 An Optimal Algorithm

3.2.1 Introductory Definitions

In this subsection we give a couple of definitions that we need in order to describe our optimal algorithm ALG$^{\rho_m}$ in Section 3.2.2, its analysis in Section 3.2.3 and a lower bound proof in Section 3.3. For algorithm ALG$^{\rho_m}$, which we provide in Section 3.4, we will use similar notations that will be given before the description of ALG$^{\rho_m}$.

As usual let $m \geq 2$ and $\mu = \{M_1, \ldots, M_m\}$ be the set of available machines. For the description of the algorithm and the attained competitive ratio we define a function $f_m(\rho)$. Intuitively, $f_m(\rho)$ represents the accumulated normalized load in a "perfect" machine profile for a subset of the jobs. In such a profile the load ratios of the first $\lfloor m/\rho \rfloor$ machines follow a Harmonic series of the form $(\rho-1)/(m-1), \ldots, (\rho-1)/(m-\lfloor m/\rho \rfloor)$ while the remaining ratios are ρ/m. Summing up these ratios we obtain $f_m(\rho)$. Formally, let

$$f_m(\rho) = (\rho-1)(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil - 1}) + \lceil (1-1/\rho)m \rceil \rho/m,$$

for any machine number $m \geq 2$ and real-valued $\rho > 1$. Here $H_k = \sum_{i=1}^{k} 1/i$ denotes the k-th Harmonic number, for any integer $k \geq 1$. We set $H_0 = 0$.

For any fixed $m \geq 2$, let ρ_m be the value satisfying $f_m(\rho_m) = 1$. Lemma 3.1 below in combination with the folklore Intermediate Value Theorem implies that ρ_m is well-defined. The algorithm we present is exactly ρ_m-competitive. By

Lemma 3.2, the values ρ_m form a non-decreasing sequence. There holds $\rho_2 = 4/3$ and $\lim_{m\to\infty} \rho_m = W_{-1}(-1/e^2)/(1+W_{-1}(-1/e^2)) \approx 1.4659$. This convergence was also stated by Englert et al. [26] but no thorough proof was presented. The two technical lemmas below are proven in Subsection 3.5.

Lemma 3.1. *The function $f_m(\rho)$ is continuous and strictly increasing in ρ, for any integer $m \geq 2$ and real number $\rho > 1$. There holds $f_m(1 + 1/(3m)) < 1$ and $f_m(2) \geq 1$.*

Lemma 3.2. *The sequence $(\rho_m)_{m\geq 2}$ is non-decreasing with $\rho_2 = 4/3$. There holds $\lim_{m\to\infty} \rho_m = W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2))$.*

Define

$$\beta(j) = \begin{cases} (\rho_m - 1)\frac{m}{m-j} & \text{if } j \leq \lfloor m/\rho_m \rfloor \\ \rho_m & \text{otherwise.} \end{cases}$$

We observe that $f_m(\rho_m) = \frac{1}{m}\sum_{j=1}^{m} \beta(j)$, taking into account that $m - \lfloor m/\rho_m \rfloor = \lceil (1 - 1/\rho_m)m \rceil$.

In order to classify jobs ALG$^{\rho_m}$ maintains a lower bound L_t on the optimum makespan. Recall that $p_t^+ = \sum_{i=1}^{t} p_i$ is the sum of the processing times of the first t jobs. For $i = 1, \ldots, 2m+1$, let p_t^i denote the processing time of the i-th largest job in $J_1, \ldots, J_t$, provided that $i \leq t$, and let $p_t^i = 0$ for $i > t$.

Obviously, when t jobs have arrived, the optimum makespan cannot be smaller than the average load $\frac{1}{m}p_t^+$ on the m machines. Moreover, the optimum makespan cannot be smaller than $3p_t^{2m+1}$, which is three times the processing time of $(2m+1)$-st largest job seen so far. Define

$$L_t = \max\{\tfrac{1}{m}p_t^+, 3p_t^{2m+1}\}.$$

For sake of completeness set $L_{n+1} := L_n$. As a shorthand let $L := L_n$.

A job $J_{t'}$, with $t' \leq t$, is called *large at time t* if $p_{t'} > (\rho_m - 1)L_t$; otherwise it is *small (at time t)*. We may say for short $J_{t'}$ is large if it is clear from context to which time t we refer. Note that, as the estimates L_t are non-decreasing over time, a job $J_{t'}$ that is large at t' does not necessarily satisfy $p_{t'} > (\rho_m - 1)L_t$ at times $t > t'$, i. e. large jobs can become small. By contrast a job J_t that is small at t is small at all times $t' > t$, i. e. small jobs remain small. Since $\rho_m \geq 4/3$, by Lemma 3.2, and $L_t \geq 3p_t^{2m+1}$, there can exist at most $2m$ jobs that are large at each time t.

Recall that we refer with $\ell(j,t)$ to the load of machine M_j at time t, which is the cumulative processing time of the jobs from $J_1, \ldots, J_{t-1}$ that were assigned to M_j. In addition to this we define $\ell_s(j,t)$ to be the total processing time of the jobs from $J_1, \ldots, J_{t-1}$ that were assigned to M_j and that are small at t.

We introduce a final piece of notation. In the sequence $p_t^1, \ldots, p_t^{2m}$ of the $2m$ largest processing times up to time t the jobs being large at t play an important role during our analysis. Thus we define for any time t and any $i = 1, \ldots, 2m$ values $\hat{p}_t^i$ as follows. We set $\hat{p}_t^i = p_t^i$ if $p_t^i > (\rho_m - 1)L_t$, and otherwise $\hat{p}_t^i = 0$.

With this let

$$L_t^* = \tfrac{1}{m}(p_t^+ - \textstyle\sum_{i=1}^{2m} \hat{p}_t^i).$$

Intuitively, L_t^* is the average machine load ignoring jobs that are large at time t. Finally, let $L^* = L_n^*$.

3.2.2 Description of the Optimal Algorithm

Algorithm ALG$^{\rho_m}$ operates in two phases, a *job arrival phase* and a *job migration phase*. In the job arrival phase all jobs of $\sigma = J_1, \ldots, J_n$ are assigned one by one to the machines. In this phase no job migrations are performed. Once σ is scheduled, the job migration phase starts. First the algorithm removes some jobs from the machines. Then these jobs are reassigned to other machines.

Job Arrival Phase. In this phase ALG$^{\rho_m}$ classifies jobs into small and large and, moreover, maintains a load profile w. r. t. the small jobs on the machines. Recall that we follow the convention that *time* t is the time when J_t has to be scheduled, for $1 \leq t \leq n$. We describe the scheduling steps in the job arrival phase. Initially, the machines are numbered in an arbitrary way and this numbering $M_1, \ldots, M_m$ remains fixed throughout the execution of ALG$^{\rho_m}$. As mentioned above the algorithm maintains a load profile on the machines as far as small jobs are concerned. Algorithm ALG$^{\rho_m}$ ensures that at any time t there exists a machine M_j satisfying $\ell_s(j,t) \leq \beta(j)L_t^*$, which is proven in Lemma 3.4.

For $t = 1, \ldots, n$, each J_t is scheduled as follows. If J_t is small, then it is scheduled on a machine with $\ell_s(j,t) \leq \beta(j)L_t^*$. If J_t is large, then it is assigned to a machine having the smallest load among all machines.

Job Migration Phase. This phase consists of a *job removal step* followed by a *job reassignment step*. At any time during this phase, let $\ell(j)$ denote the current load of M_j, $1 \leq j \leq m$. In the removal step ALG$^{\rho_m}$ maintains a set σ_R of removed jobs. Initially $\sigma_R = \emptyset$. During the removal step, while there exists a machine M_j whose load $\ell(j)$ exceeds $\max\{\beta(j)L^*, (\rho_m - 1)L\}$, ALG$^{\rho_m}$ removes the job with the largest processing time currently residing on M_j and adds the job to σ_R.

If $\sigma_R = \emptyset$ at the end of the removal step, then ALG$^{\rho_m}$ terminates. If $\sigma_R \neq \emptyset$, then the reassignment step is executed. Let $\sigma_R' \subseteq \sigma_R$ be the subset of the jobs that are large at the end of σ, i. e. whose processing time is greater than $(\rho_m - 1)L$. Again there can exist at most $2m$ such jobs. ALG$^{\rho_m}$ first sorts the jobs of σ_R'

in order of non-increasing processing time; ties are broken arbitrarily. Let J_r^i, $1 \leq i \leq |\sigma'_R|$, be the i-th job in this sorted sequence and p_r^i be its processing time. For $i = 1, \ldots, m$, ALG^{ρ_m} forms job pairs consisting of the i-th largest and the $(2m+1-i)$-th largest jobs provided that the processing time of the latter job is sufficiently high. A pairing strategy combining the i-th largest and the $(2m+1-i)$-th largest jobs was also used by Graham [35]. Formally, ALG^{ρ_m} builds sets $\sigma_1, \ldots, \sigma_m$ that contain up to two jobs. Initially, all these sets are empty. In a first step J_r^i is assigned to σ_i, for any i with $1 \leq i \leq \min\{m, |\sigma'_R|\}$. In a second step J_r^{2m+1-i} is added to σ_i provided that $p_r^{2m+1-i} > p_r^i/2$, i. e. the processing time of J_r^{2m+1-i} must be greater than half times that of J_r^i. This second step is executed for any i such that $1 \leq i \leq m$ and $2m+1-i \leq |\sigma'_R|$. For any set σ_i, $1 \leq i \leq m$, let $p(\sigma_i) = \sum_{t:J_t \in \sigma_i} p_t$ be the total processing time of the jobs in σ_i. ALG^{ρ_m} now renumbers the sets in non-increasing order of total processing times such that $p(\sigma_1) \geq \ldots \geq p(\sigma_m)$. Then, for $i = 1, \ldots, m$, it takes the set σ_i and assigns the jobs of σ_i to a machine with the smallest current load. If σ_i contains two jobs, then both are placed on the same machine. Finally, if $\sigma_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m) \neq \emptyset$, then ALG^{ρ_m} takes care of the remaining jobs. These jobs may be scheduled in an arbitrary order. Each job of $\sigma_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$ is scheduled on a machine having the smallest current load. This concludes the description of ALG^{ρ_m}. A summary in pseudo-code is given in Figure 3.1.

As we will see in the analysis of ALG^{ρ_m}, in the job migration phase the algorithm has to remove at most $\varphi_m = \lceil (2-\rho_m)/(\rho_m-1)^2 \rceil + 4$ jobs from each machine. Table 3.1 depicts the competitive ratios ρ_m (exactly and approximately) and the migration numbers φ_m, for small values of m. We point out that ρ_m is a rational number, for any $m \geq 2$.

m	2	3	4	5	6
ρ_m	$\frac{4}{3}$	$\frac{15}{11}$	$\frac{11}{8}$	$\frac{125}{89}$	$\frac{137}{97}$
$\approx$		1.3636	1.3750	1.4045	1.4124
φ_m	10	9	9	8	8

m	7	8	9	10	11
ρ_m	$\frac{273}{193}$	$\frac{586}{411}$	$\frac{1863}{1303}$	$\frac{5029}{3517}$	$\frac{58091}{40451}$
$\approx$	1.4145	1.4258	1.4298	1.4299	1.4360
φ_m	8	8	8	8	7

Table 3.1: The values of ρ_m and φ_m, for small m.

In summary, our result is the following.

Theorem 3.3. *Algorithm* ALG$^{\rho_m}$ *is* ρ_m*-competitive and uses no more than* $(\lceil(2-\rho_m)/(\rho_m-1)^2\rceil+4)m$ *job migrations.*

Job arrival phase.

- Each J_t, $1 \le t \le n$, is scheduled as follows.
  - If J_t is small, then assign J_t to an M_j with $\ell_s(j,t) \le \beta(j)L_t^*$.
  - If J_t is large, then assign J_t to a least loaded machine.

Job migration phase.

- Job removal:
  - Set $\sigma_R := \emptyset$.
  - While there exists an M_j with $\ell(j) > \max\{\beta(j)L^*, (\rho-1)L\}$
    - remove the largest job J_l from M_j
    - add J_l to σ_R.
- Job reassignment:
  - Let $\sigma'_R = \{J_i \in \sigma_R \mid p_i > (\rho_m - 1)L\}$.
  - For $i = 1, \ldots, m$ do
    - initialize $\sigma_i = \emptyset$.
    - if $i \le |\sigma'_R|$, then add J_r^i to σ_i.
    - if $2m+1-i \le |\sigma'_R|$ and $p_r^{2m+1-i} > p_r^i/2$, then add J_r^{2m+1-i} to σ_i.
  - Enumerate the sets σ_i non-increasingly by total processing time.
  - For $i = 1, \ldots, m$, assign σ_i to a least loaded machine.
  - Assign each $J_i \in \sigma_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$ to a least loaded machine.

Figure 3.1: Algorithm ALG$^{\rho_m}$.

3.2.3 Analysis of the Optimal Algorithm

We first show that the assignment operations in the job arrival phase are well defined. A corresponding statement was shown by Englert et al. [26]. The following

proof is more involved because we have to take care of large jobs in the current schedule.

Lemma 3.4. *At any time t there exists a machine M_j satisfying $\ell_s(j,t) \leq \beta(j)L_t^*$.*

Proof. Suppose that there exists a time t, $1 \leq t \leq n$, such that $\ell_s(j,t) > \beta(j)L_t^*$ holds for all M_j, $1 \leq j \leq m$. We will derive a contradiction.

Among the jobs $J_1, \ldots, J_t$, at most $2m$ can be large at time t: If there were at least $2m+1$ such jobs, then $L_t \geq 3p_t^{2m+1} > 3(\rho_m - 1)L_t \geq L_t$ because $\rho_m \geq 4/3$, see Lemma 3.2. Hence each of the jobs that is large at time t is represented by a positive entry in the sequence $\hat{p}_t^1, \ldots, \hat{p}_t^{2m}$. Conversely, every positive entry in this sequence corresponds to a job that is large at time t and resides on one of the m machines or is equal to J_t if J_t is large. Hence if J_t is large, $p_t^+ = \sum_{j=1}^m \ell(j,t) + p_t = \sum_{j=1}^m \ell_s(j,t) + \sum_{i=1}^{2m} \hat{p}_t^i$. If J_t is small, then $p_t^+ = \sum_{j=1}^m \ell(j,t) + p_t \geq \sum_{j=1}^m \ell(j,t) = \sum_{j=1}^m \ell_s(j,t) + \sum_{i=1}^{2m} \hat{p}_t^i$. In either case

$$\begin{aligned} p_t^+ &= \sum_{j=1}^{m} \ell(j,t) + p_t \geq \sum_{j=1}^{m} \ell_s(j,t) + \sum_{i=1}^{2m} \hat{p}_t^i > \sum_{j=1}^{m} \beta(j)L_t^* + \sum_{i=1}^{2m} \hat{p}_t^i \\ &= m(\rho_m - 1)L_t^* \sum_{j=1}^{\lfloor m/\rho_m \rfloor} 1/(m-j) + (m - \lfloor m/\rho_m \rfloor)\rho_m L_t^* + \sum_{i=1}^{2m} \hat{p}_t^i. \end{aligned}$$

Taking into account that $m - \lfloor m/\rho_m \rfloor = \lceil (1-1/\rho_m)m \rceil$ and that $f_m(\rho_m) = 1$, we obtain

$$\begin{aligned} p_t^+ = \sum_{j=1}^{m} \ell(j,t) + p_t &> mL_t^* \big((\rho_m - 1)(H_{m-1} - H_{\lceil (1-1/\rho_m)m \rceil - 1}) \\ &\quad + \lceil (1 - 1/\rho_m)m \rceil \rho_m / m \big) + \sum_{i=1}^{2m} \hat{p}_t^i \\ &= mL_t^* f_m(\rho_m) + \sum_{i=1}^{2m} \hat{p}_t^i \\ &= m(1/m \sum_{i=1}^{t} p_t - 1/m \sum_{i=1}^{2m} \hat{p}_t^i) + \sum_{i=1}^{2m} \hat{p}_t^i \;=\; \sum_{i=1}^{t} p_i = p_t^+, \end{aligned}$$

which is a contradiction. □

We next analyze the job migration phase. Firstly we bound the number of job removals.

Lemma 3.5. *In the job removal step* ALG^{ρ_m} *removes at most* $\varphi_m = \lceil (2-\rho_m)/(\rho_m-1)^2 \rceil + 4$ *jobs from each of the machines.*

Proof. Recall that time $n+1$ is the time when the entire job sequence σ is scheduled and the job migration phase with the removal step starts. Consider any M_j, with $1 \leq j \leq m$. We show that it suffices to remove at most $\varphi_m = \lceil (2-\rho_m)/(\rho_m-1)^2 \rceil + 4$ jobs at time $n+1$ so that M_j's resulting load is upper bounded by $\max\{\beta(j)L^*, (\rho_m-1)L\}$. Since ALG^{ρ_m} always removes the largest jobs, the lemma follows.

First assume $\ell_s(j, n+1) \leq \beta(j)L^*$. If at time $n+1$ machine M_j does not contain any jobs that are large, then $\ell(j, n+1) = \ell_s(j, n+1) \leq \beta(j)L^*$. In this case no job has to be removed and we are done. If M_j does contain jobs that are large at time $n+1$, then it suffices to remove these jobs. Let time l be the last time when a job J_l that is large at time $n+1$ was assigned to M_j. Since $L_l \leq L$, J_l was also large at time l and hence it was assigned to a least loaded machine. This implies that prior to the assignment of J_l, M_j has a load of at most $p_l^+/m \leq L_l \leq L$. Hence it could contain at most $1/(\rho_m-1)$ jobs that are large at time $n+1$ because any such job has a processing time greater than $(\rho_m-1)L$. Hence at most $1/(\rho_m-1)+1$ jobs have to be removed from M_j, and the latter expression is upper bounded by φ_m.

Next assume $\ell_s(j, n+1) > \beta(j)L^*$. If $\ell_s(j, n) \leq \beta(j)L^* = \beta(j)L_n^*$, then J_n was assigned to M_j. In this case it suffices to remove J_n and, as in the previous case, at most $1/(\rho_m-1)+1$ jobs that are large at time $n+1$. Again $1/(\rho_m-1)+2 \leq \varphi_m$.

In the remainder of this proof we consider the case that $\ell_s(j, n+1) > \beta(j)L^*$ and $\ell_s(j, n) > \beta(j)L^*$. Let t^* be the earliest time such that $\ell_s(j, t) > \beta(j)L_t^*$ holds for all times $t^* \leq t \leq n$. We have $t^* \geq 2$ because $\ell_s(j, 1) = 0 \leq \beta(j)L_1^*$. Hence time t^*-1 exists. We partition the jobs residing on M_j at time $n+1$ into three sets. Set T_1 is the set of jobs that were assigned to M_j at or before time t^*-1 and are small at time t^*-1. Set T_2 contains the jobs that were assigned to M_j at or before time t^*-1 and are large at time t^*-1. Finally T_3 is the set of jobs assigned to M_j at or after time t^*. We show a number of claims that we will use in the further proof.

Claim 3.5.1. Each job in $T_2 \cup T_3$ is large at the time it is assigned to M_j.

Claim 3.5.2. There holds $\sum_{J_i \in T_1 \setminus \{J_l\}} p_i \leq \beta(j)L_{t^*-1}^*$, where J_l is the job of T_1 that was assigned last to M_j.

Claim 3.5.3. There holds $|T_2| \leq 3$.

Claim 3.5.4. For any $J_l \in T_3$, M_j's load immediately before the assignment of J_l is at most L_l.

Claim 3.5.5. Let $J_l \in T_3$ be the last job assigned to M_j. If M_j contains at least $\varphi_m - 4$ jobs, different from J_l, each having a processing time of at least $(\rho_m - 1)^2 L$, then it suffices to remove these $\varphi_m - 4$ jobs and J_l such that M_j's resulting load is upper bounded by $(\rho_m - 1)L$.

Claim 3.5.6. If there exists a $J_l \in T_3$ with $p_l < (\rho_m - 1)^2 L$, then M_j's load immediately before the assignment of J_l is at most $(\rho_m - 1)L$.

Proof of Claim 3.5.1. The jobs of T_2 are large at time $t^* - 1$ and hence at the time they were assigned to M_j. By the definition of t^*, $\ell_s(j,t) > \beta(j)L_t^*$ for any $t^* \leq t \leq n$. Hence ALG^{ρ_m} does not assign small jobs to M_j at or after time t^*.

Proof of Claim 3.5.2. All jobs of $T_1 \setminus \{J_l\}$ are small at time $t^* - 1$ and their total processing time is at most $\ell_s(j, t^* - 1)$. In fact, their total processing time is equal to $\ell_s(j, t^*-1)$ if $l = t^*-1$. By the definition of t^*, $\ell_s(j, t^*-1) \leq \beta(j)L^*_{t^*-1}$.

Proof of Claim 3.5.3. We show that for any time t, $1 \leq t \leq n$, when J_t has been placed on a machine, M_j can contain at most three jobs that are large at time t. The claim then follows by considering $t^* - 1$. Suppose that when J_t has been scheduled, M_j contained more than three jobs that are large at time t. Among these jobs let J_l be the one that was assigned last to M_j. Immediately before the assignment of J_l machine M_j had a load greater than L_l because the total processing time of three large jobs is greater than $3(\rho_m-1)L_t \geq 3(\rho_m-1)L_l \geq L_l$ since $\rho_m \geq 4/3$, see Lemma 3.2. This contradicts the fact that J_l is placed on a least loaded machine, which has a load of at most $p_l^+/m \leq L_l$.

Proof of Claim 3.5.4. By Claim 3.5.1 J_l is large at time l and hence is assigned to a least loaded machine, which has a load of at most $p_l^+/m \leq L_l$.

Proof of Claim 3.5.5. Claim 3.5.4 implies that immediately before the assignment of J_l machine M_j has a load of at most $L_l \leq L$. If M_j contains at least $\varphi_m - 4$ jobs, different from J_l, with a processing time of at least $(\rho_m - 1)^2 L$, then the removal of these $\varphi_m - 4$ jobs and J_l from M_j leads to a machine load of at most $L - (\varphi_m - 4)(\rho_m - 1)^2 L \leq L - \lceil (2 - \rho_m)/(\rho_m - 1)^2 \rceil (\rho_m - 1)^2 L \leq (\rho_m - 1)L$, as desired.

Proof of Claim 3.5.6. By Claim 3.5.1 J_l is large at time l and hence $p_l > (\rho_m - 1)L_l$. Since $p_l < (\rho_m - 1)^2 L$, it follows $L_l < (\rho_m - 1)L$. By Claim 3.5.4, M_j's load prior to the assignment of J_l is at most L_l and hence at most $(\rho_m - 1)L$.

We now finish the proof of the lemma and distinguish two cases depending on the cardinality of $T_2 \cup T_3$.

Case 1: If $|T_2 \cup T_3| < \varphi_m$, then by Claim 3.5.2 it suffices to remove the jobs of $T_2 \cup T_3$ and the last job of T_1 assigned to M_j.

Case 2: Suppose $|T_2 \cup T_3| \geq \varphi_m$. By Claim 3.5.3, $|T_2| \leq 3$ and hence $|T_3| \geq \varphi_m - 3$. Among the jobs of T_3 consider the last $\varphi_m - 3$ ones assigned to M_j. If each of them has a processing time of at least $(\rho_m - 1)^2 L$, then Claim 3.5.5

ensures that it suffices to remove these $\varphi_m - 3$ jobs. If one of them, say J_l, has a processing time smaller than $(\rho_m - 1)^2 L$, then by Claim 3.5.6 M_j's load prior to the assignment of J_l is at most $(\rho_m - 1)L$. Again it suffices to remove these $\varphi_m - 3$ jobs from M_j. □

We now show that after the job reassignment step the makespan of ALG^{ρ_m} is bounded by $\rho_m \text{OPT}$, provided that it was bounded by this value after the job removal phase.

Lemma 3.6. *If $\ell(j) \leq \rho_m \text{OPT}$ for all $1 \leq j \leq m$ after the job removal step, then $\ell(j) \leq \rho_m \text{OPT}$ for all $1 \leq j \leq m$ after the reassignment step.*

Proof. We show that all scheduling operations in the reassignment step preserve a load of at most $\rho_m \text{OPT}$ on each of the machines. We first consider the assignment of the sets $\sigma_1, \ldots, \sigma_m$. Suppose that these sets are already sorted in order of non-increasing total processing times, i. e. $p(\sigma_1) \geq \ldots \geq p(\sigma_m)$. We first argue that $p(\sigma_1)$ and hence every $p(\sigma_i)$, $1 \leq i \leq m$, is upper bounded by OPT. If σ_1 contains at most one job, there is nothing to show because OPT cannot be smaller than the processing time of any job in σ. Assume that σ_1 contains two jobs. Then it consists of jobs $J_r^{i_1}$ and $J_r^{2m+1-i_1}$, for some i_1 with $1 \leq i_1 \leq m$. Since the two jobs are paired there holds $p_r^{2m+1-i_1} > p_r^{i_1}/2$ and hence $p_r^{2m+1-i_1} > p(\sigma_1)/3$. Let OPT' denote the optimum makespan for the job sequence $J_r^1, \ldots, J_r^{2m+1-i_1}$. Since $J_r^{i_1}$ and $J_r^{2m+1-i_1}$ are paired, jobs J_r^i and J_r^{2m+1-i} are also paired, for any $i_1 < i \leq m$, because $p_r^{2m+1-i} \geq p_r^{2m+1-i_1} > p_r^{i_1}/2 \geq p_r^i/2$. Hence the sets $\sigma_1, \ldots, \sigma_m$ contain all the jobs $J_r^1, \ldots, J_r^{2m+1-i_1}$, which implies $p(\sigma_1) \geq \text{OPT}'$ and $p_r^{2m+1-i_1} > \text{OPT}'/3$. It follows $p_r^i > \text{OPT}'/3$, for all i with $1 \leq i \leq 2m + 1 - i_1$. Graham [35] showed that given a sequence of up to $2m$ jobs, each having a processing time greater than a third times the optimum makespan, an optimal schedule is obtained by repeatedly pairing the i-th largest and $(2m + 1 - i)$-th largest jobs of the sequence. This is exactly the assignment computed by ALG^{ρ_m} for $J_r^1, \ldots, J_r^{2m+1-i_1}$. We conclude $p(\sigma_1) = \text{OPT}'$ and $p(\sigma_1) \leq \text{OPT}$.

A final observation is that each job of σ'_R that is not contained in $\sigma_1 \cup \ldots \cup \sigma_m$ has a processing time of at most $\text{OPT}/3$. A job in $\sigma'_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$ is equal to a job $J_r^{2m+1-i_0}$, with $1 \leq i_0 \leq m$. Since $J_r^{2m+1-i_0}$ is not paired with $J_r^{i_0}$, there holds $p_r^{2m+1-i_0} \leq p_r^{i_0}/2$. Assume that $p_r^{2m+1-i_0} > \text{OPT}/3$. Then $p_r^{2m+1-i_0}$ is greater than a third times the optimum makespan for the jobs $J_r^1, \ldots, J_r^{2m+1-i_0}$. Using again the results by Graham [35], we find an optimal schedule for the latter job sequence by repeatedly pairing J_r^i with J_r^{2m+1-i}. However, since $p_r^{2m+1-i_0} \leq p_r^{i_0}/2$, the processing time $p_r^{2m+1-i_0}$ is at most a third times the resulting optimum makespan for $J_r^1, \ldots, J_r^{2m+1-i_0}$. Hence $p_r^{2m+1-i_0}$ is at most a third times OPT, which is a contradiction.

Next we compare the processing time of the jobs of $\sigma_1 \cup \ldots \cup \sigma_m$ to $\sum_{i=1}^{2m} \hat{p}_n^i$. Set σ'_R contains the jobs of σ_R that are large at time n. There exist at most $2m$ jobs that are large at time n and hence the processing time of each job in σ'_R is represented by a positive entry in the sequence $\hat{p}_n^1, \ldots, \hat{p}_n^{2m}$. It follows that the total processing time of the jobs in σ'_R and hence the total processing time of the jobs in $\sigma_1 \cup \ldots \cup \sigma_m$ is at most $\sum_{i=1}^{2m} \hat{p}_n^i$. Recall that $p(\sigma_1) \geq \ldots \geq p(\sigma_m)$. Then, for any j with $1 \leq j \leq m$, the product $jp(\sigma_j)$ is upper bounded by the total processing time of $\sigma_1 \cup \ldots \cup \sigma_m$ and hence $jp(\sigma_j) \leq \sum_{i=1}^{2m} \hat{p}_n^i$.

Now consider the assignment of the sets $\sigma_1, \ldots, \sigma_m$ to the machines. Each set is assigned to a least loaded machine. Hence when σ_j, $1 \leq j \leq m$, is scheduled, it is assigned to a machine whose current load is at most $\max\{\beta(j)L^*, (\rho_m - 1)L\}$. If the load is at most $(\rho_m - 1)L$, then the machine's load after the assignment is at most $(\rho_m - 1)L + p(\sigma_j) \leq (\rho_m - 1)L + \text{OPT} \leq \rho_m \text{OPT}$. If the current load is only upper bounded by $\beta(j)L^*$, then we distinguish two cases.

If $j \leq \lfloor m/\rho_m \rfloor$, then $j \leq m/\rho_m$, which is equivalent to $m/(m-j) \leq \rho_m/(\rho_m - 1)$. The resulting machine load is at most

$$\begin{aligned}\beta(j)L^* + p(\sigma_j) &= (\rho_m - 1)\frac{m}{m-j}\left(\frac{1}{m}\sum_{i=1}^{n} p_i - \frac{1}{m}\sum_{i=1}^{2m} \hat{p}_n^i\right) + p(\sigma_j) \\ &\leq (\rho_m - 1)\frac{1}{m-j}(mL - jp(\sigma_j)) + p(\sigma_j).\end{aligned}$$

The last inequality follows because, as argued above, $jp(\sigma_j) \leq \sum_{i=1}^{2m} \hat{p}_n^i$. It follows that the machine load is upper bounded by

$$(\rho_m - 1)\tfrac{1}{m-j}(mL - mp(\sigma_j)) + \rho_m p(\sigma_j) \leq \rho_m(L - p(\sigma_j)) + \rho_m p(\sigma_j) = \rho_m L.$$

The last inequality holds because $m/(m-j) \leq \rho_m/(\rho_m - 1)$, as stated above.

If $j > \lfloor m/\rho_m \rfloor$, then $j \geq m/\rho_m$ because j is integral. In this case the machine load is upper bounded by

$$\begin{aligned}\beta(j)L^* + p(\sigma_j) &= \rho_m\left(\sum_{i=1}^{n} p_i - \sum_{i=1}^{2m} \hat{p}_n^i\right)/m + p(\sigma_j) \\ &\leq \rho_m\left(\sum_{i=1}^{n} p_i - jp(\sigma_j)\right)/m + p(\sigma_j) \;\leq\; \rho_m L\end{aligned}$$

because $j\rho_m \geq m$.

Finally we consider the jobs $\sigma_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$. Each job of $\sigma_R \setminus \sigma'_R$ has a processing time of at most $(\rho_m - 1)L$. As argued above, each job of $\sigma'_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$ has a processing time of at most $\text{OPT}/3$, which is upper bounded by

$(\rho_m - 1)\text{OPT}$ since $\rho_m \geq 4/3$. Hence each job of $\sigma_R \setminus (\sigma_1 \cup \ldots \cup \sigma_m)$ has a processing time of at most $(\rho_m - 1)\text{OPT}$. Each of these jobs is scheduled on a least loaded machine and thus after the assignment the corresponding machine has a load of at most $\text{OPT} + (\rho_m - 1)\text{OPT} \leq \rho_m \text{OPT}$. □

We now can prove Theorem 3.3.

Proof. (*of Theorem 3.3).* Lemma 3.4 establishes that the assignment operations of algorithm ALG^{ρ_m} in the job arrival phase are well-defined and each job is scheduled to a machine M_j. By the algorithm, after the job removal step each machine M_j, $1 \leq j \leq m$, has a load of at most $\max\{\beta(j)L^*, (\rho_m - 1)L\}$. We observe that this load is at most $\rho_m L$. If $(\rho_m - 1)L \geq \beta(j)L^*$, there is nothing to show. We evaluate $\beta(j)L^*$. If $j > \lfloor m/\rho_m \rfloor$, then $\beta(j) = \rho_m$ and $\beta(j)L^* = \rho_m L^* \leq \rho_m L$. If $j \leq \lfloor m/\rho_m \rfloor$, then $\beta(j) = (\rho_m - 1)m/(m - j) \leq (\rho_m - 1)m/(m - \lfloor m/\rho_m \rfloor) = (\rho_m - 1)m/\lceil (1 - 1/\rho_m)m \rceil) \leq \rho_m$ and thus $\beta(j)L^* \leq \rho_m L$. Hence M_j's load is upper bounded by $\rho_m \text{OPT}$. Lemma 3.6 ensures that, after the reassignment step, each machine still has a load of at most $\rho_m \text{OPT}$. This already gives the claim on the competitive ratio of ALG^{ρ_m}. Lemma 3.5 settles that the number of job migrations is bounded as desired. The proof of the theorem is complete. □

3.3 Lower Bounds

We present two lower bounds showing that ALG^{ρ_m} is optimal. The next theorem shows that a competitive ratio of smaller than ρ_m can not be attained using $o(n)$ migration.

Theorem 3.7. *Let $m \geq 2$. No deterministic online algorithm can achieve a competitive ratio smaller than ρ_m if $o(n)$ job migrations are allowed.*

Proof. Let ALG be any deterministic online algorithm that is allowed to use up to $g(n)$ job migrations on a job sequence of length n. Suppose there is an $\varepsilon > 0$ such that ALG achieves a competitive ratio strictly smaller than $\rho_m - \varepsilon < \rho_m$. We will derive a contradiction.

Choose $\varepsilon' = \varepsilon/3$. Since $g(n) = o(n)$ there exists an n_0 such that $g(n)/n \leq \varepsilon'/(2m)$, for all $n \geq n_0$. Hence there exists an n_0 such that $g(n+m)/(n+m) \leq \varepsilon'/(2m)$, for all $n \geq \max\{m, n_0\}$. Let n', with $n' \geq \max\{m, n_0\}$, be the smallest integer multiple of m. Because $n' + m \leq 2n'$, we have $g(n'+m)/n' \leq \varepsilon'/m$. An adversary constructs a job sequence consisting of $n' + m$ jobs. Let $s_1 = m/n'$. By our choice of n', there holds $s_1 \leq \varepsilon'/g(n'+m)$. The following adversarial sequence is similar to that used by Englert et al. [26]. However, here we have to ensure that an online algorithm cannot benefit much from migrating $o(n)$ jobs.

First the adversary presents n' jobs of processing time s_1. If after the assignment of these jobs ALG has a machine M_j, $1 \leq j \leq m$, whose load is at least ρ_m, then the adversary presents m jobs of processing time $s_2 = \varepsilon'/m$. Using job migration, ALG can remove at most $g(n'+m)$ jobs of processing time s_1 from M_j. Since $g(n'+m)s_1 \leq \varepsilon'$, after job migration M_j still has a load of at least $\rho_m - \varepsilon'$. On the other hand the optimal makespan is $1 + \varepsilon'/m$. In an optimal assignment each machine contains n'/m jobs of processing time s_1 and one job with processing time s_2. The ratio $(\rho_m - \varepsilon')/(1 + \varepsilon'/m)$ is at least $\rho_m - \varepsilon$ by our choice of ε' and the fact that $\rho_m \leq 2$, see Lemma 3.1. We obtain a contradiction.

In the following we study the case that after the assignment of the jobs with processing time s_1 each machine in ALG's schedule has a load strictly smaller than ρ_m. We number the machines in order of non-decreasing load such that $\ell(1) \leq \ldots \leq \ell(m)$. Here $\ell(j)$ denotes the load of M_j after the jobs with processing time s_1 have arrived, $1 \leq j \leq m$. For $j = 1, \ldots, m-1$, define $\beta(j) = (\rho_m - 1)m/(m-j)$. We first argue that there must exist a machine M_j, $1 \leq j \leq m-1$, in ALG's schedule whose load is at least $\beta(j)$. Suppose that each machine M_j, $1 \leq j \leq m-1$, had a load strictly smaller than $\beta(j)$. By Lemma 3.1, $\rho_m > 1$ and hence $\lceil (1-1/\rho_m)m \rceil \geq 1$. Consider the $\lceil (1-1/\rho_m)m \rceil$ machines with the highest load in ALG's schedule. Each of these machines has a load strictly smaller than ρ_m. The remaining machines have a load strictly smaller than $\beta(j) = (\rho_m - 1)m/(m-j)$, for $j = 1, \ldots, m - \lceil (1-1/\rho_m)m \rceil$. We conclude that after the arrival of the jobs with processing time s_1 the total load on the machines is strictly smaller than

$$
\begin{aligned}
&(\rho_m - 1)m \sum_{j=1}^{m-\lceil (1-1/\rho_m)m \rceil} \frac{1}{m-j} + \lceil (1-1/\rho_m)m \rceil \rho_m \\
&= m((\rho_m - 1)(H_{m-1} - H_{\lceil (1-1/\rho_m)m \rceil - 1}) + \lceil (1-1/\rho_m)m \rceil \rho_m / m) \\
&= m f_m(\rho_m) = m.
\end{aligned}
$$

The last equation holds because $f_m(\rho_m) = 1$, by the choice of ρ_m. We obtain a contradiction to the fact that after the arrival of the jobs with processing time s_1 a total load of exactly m resides on the machines.

Let M_{j_0}, with $1 \leq j_0 \leq m-1$, be a machine whose load is at least $\beta(j_0)$. Since ALG's machines are numbered in order of non-decreasing load there exist at most $j_0 - 1$ machines having a smaller load than $\beta(j_0)$. The adversary presents j_0 jobs of processing time $s_2 = m/(m-j_0)$. Using job migration ALG can remove at most $g(n'+m)$ job with processing time s_1 from any of the machines, thereby reducing the load by at most ε'. Hence in ALG's final schedule there exists a machine having a load of a least $\beta(j_0) + m/(m-j_0) - \varepsilon'$. This holds true if the jobs with processing time s_2 reside on different machines. If there exists a machine

containing two jobs of processing time s_2, then its load is at least $2m/(m-j_0) \geq (\rho_m - 1)m/(m-j_0) + m/(m-j_0) = \beta(j_0) + m/(m-j_0) = \rho_m m/(m-j_0)$ as desired. The inequality holds because $\rho_m \leq 2$, by Lemma 3.1. Hence ALG's makespan is at least $\beta(j_0) + m/(m-j_0) - \varepsilon'$.

The optimum makespan for the job sequence is upper bounded by $m/(m-j_0)+\varepsilon'$. In an optimal schedule the j_0 jobs having processing time s_2 are assigned to different machines. The n' many jobs of processing time s_1 are distributed evenly among the remaining $m - j_0$ machines. If n' is an integer multiple of $m - j_0$, then the load on each of these $m - j_0$ machines is exactly $n's_1/(m - j_0) = m/(m - j_0)$, which is exactly equal to s_2. If n' is not divisible by $m - j_0$, then the maximum load on any of these $m - j_0$ machines cannot be higher than $m/(m - j_0) + s_1 \leq m/(m - j_0) + \varepsilon'/g(n' + m) \leq m/(m - j_0) + \varepsilon'$.

Dividing the lower bound on ALG's makespan by the upper bound on the optimum makespan we obtain $(\rho_m m/(m - j_0) - \varepsilon')/(m/(m - j_0) + \varepsilon') \geq (\rho_m - \varepsilon')/(1 + \varepsilon') \geq \rho_m - \varepsilon$. The last inequality holds because $\varepsilon' = \varepsilon/3$ and $\rho_m \leq 2$, see Lemma 3.1. We obtain a contradiction to the assumption that ALG's competitiveness is strictly smaller than $\rho_m - \varepsilon$. □

We present a lower bound that states that a migration of $\Omega(m)$ is really necessary to obtain a competitiveness at most $1 + 1/\sqrt{2} \approx 1.707$. This shows that no algorithm can use $o(m)$ migration to achieve the competitiveness of our optimal algorithm. Additionally it shows that our family of algorithms has to use migration $\Omega(m)$ to obtain a competitiveness better than $1 + 1/\sqrt{2}$. The job sequence is borrowed from [28]. It is also used in [26]. However, the argumentation on the number of migrations needed is different here.

Theorem 3.8. *Every deterministic algorithm* ALG *that reassigns less than* $\lfloor m/12 \rfloor$ *many jobs has a competitive ratio* $\rho(\text{ALG}) \geq 1 + 1/\sqrt{2} > 1.707$.

Proof. Let ALG be any deterministic algorithm that reassign less than $\lfloor m/12 \rfloor$ many jobs. We show that ALG has a competitive ratio of at least $1 + 1/\sqrt{2}$. An adversary presents a sequence of m jobs with processing time $s_1 = 1$ each. If, after these jobs are placed on their machines, ALG's schedule contains less than $m - \lfloor m/12 \rfloor$ many machines containing at least one job of processing time s_1, then the adversary presents no more jobs. It follows, ALG's competitive ratio is at least two if it performs less than $\lfloor m/12 \rfloor$ many job reassignments, which can be seen as follows. If there are less than $m - \lfloor m/12 \rfloor$ machines with a load at least s_1, then, as long as less than $\lfloor m/12 \rfloor$ jobs are removed, each from a most loaded machine, there has to be a machine with load larger than s_1 since otherwise this yields a contradiction to the fact that m jobs of processing time s_1 were assigned to the machines.

Now assume that ALG's schedule contains at least $m - \lfloor m/12 \rfloor$ machines with a load at least s_1. The adversary reveals m jobs of processing time $s_2 = 1+\sqrt{2}$. If, after ALG has placed these jobs on the machines, there exist less than $m-2\lfloor m/12 \rfloor$ many machines with load at least $s_1 + s_2$, then the adversary does not present any further jobs. Again, if there are less than $m - 2\lfloor m/12 \rfloor$ many machines with load at least $s_1 + s_2$ in ALG's schedule, then there must be a machine with load at least s_1+2s_2 as long as less than $m-\lfloor m/12 \rfloor$ jobs are removed from its schedule. This holds true because of the following. Before the jobs with processing time s_2 were assigned to the machines, at least $m - \lfloor m/12 \rfloor$ many machines had a load of at least s_1. Hence at most $\lfloor m/12 \rfloor$ machines were empty. Thus at least $m - \lfloor m/12 \rfloor$ many jobs having processing time s_2 have to be assigned to a machine that has load at least s_1, and then the load of the respective machine is at least $s_1 + s_2$ after the assignment. Consequently, if there exist less than $m - 2\lfloor m/12 \rfloor$ many machines with load at least $s_1 + s_2$ and less than $\lfloor m/12 \rfloor$ jobs are removed, there remains a machine in ALG's schedule with load at least $s_1 + 2s_2 = 1 + 2(1 + \sqrt{2})$. The optimum makespan on the presented sequence is no more than $s_1 + s_2$ and thus the competitive ratio of ALG is at least $1 + 1/\sqrt{2}$.

Now consider the case that ALG's schedule contains at least $m - 2\lfloor m/12 \rfloor$ many machines with load at least $s_1 + s_2$. The adversary presents $\lfloor m/4 \rfloor$ many jobs with processing time $s_3 = 2 + \sqrt{2}$. As there are at least $m - 2\lfloor m/12 \rfloor$ many machines with load at least $s_1 + s_2$, it follows that at least $\lfloor m/4 \rfloor - 2\lfloor m/12 \rfloor$ many jobs with processing time s_3 have to be assigned to a machine with load at least $s_1 + s_2$. Hence after the removal of less than $\lfloor m/12 \rfloor$ many jobs ALG's makespan is still at least $s_1 + s_2 + s_3$ since there still remains a machine with load at least $s_1 + s_2 + s_3$. In an optimal schedule the makespan is no larger than s_3. The jobs with processing time s_3 reside each on their own a machine. Two jobs of processing time s_2 are paired, and if m is not divisible by two, then the remaining job with processing time s_2 is paired with one job having processing time s_1. To each of the remaining empty machines four jobs of processing time s_1 are assigned or less if there are less than four jobs with processing time s_1 left to be reassigned. It follows ALG has a competitive ratio of at least $1 + 1/\sqrt{2}$. □

3.4 Algorithms Using Fewer Migrations

We present a family of algorithms $\text{ALG}^{\hat{\rho}}$ that uses a smaller number of job migrations. We first describe the family and then analyze its performance.

3.4.1 Description of $\text{ALG}^{\hat{\rho}}$

Algorithm $\text{ALG}^{\hat{\rho}}$ is defined for any constant $\hat{\rho}$ with $5/3 \leq \hat{\rho} \leq 2$, where $\hat{\rho}$ is the targeted competitive ratio. An important feature of $\text{ALG}^{\hat{\rho}}$ is that it par-

titions the machines $M_1, \ldots, M_m$ into two sets $\mu_A = \{M_1, \ldots, M_{\lfloor m/2 \rfloor}\}$ and $\mu_B = \{M_{\lfloor m/2 \rfloor + 1}, \ldots, M_m\}$ of roughly equal size. In a job arrival phase the jobs are preferably assigned to machines in μ', provided that their load it not too high. In the job migration phase, jobs are mostly migrated from machines of μ_A, preferably to machines in μ_B, and this policy will allow us to achieve a smaller number of migrations. Setting $\hat{\rho} = 5/3$, we obtain an algorithm ALG$^{\hat{\rho}}$ that is $5/3$-competitive using $4m$ migrations. For $\hat{\rho} = 1.75$ the resulting algorithm ALG$^{\hat{\rho}}$ is 1.75-competitive and uses at most $2.5m$ migrations. In the following let $5/3 \leq \hat{\rho} \leq 2$. Again our algorithm consists of a job arrival phase and a job migration phase. We describe ALG$^{\hat{\rho}}$ informally.

Job Arrival Phase. At any time t algorithm ALG$^{\hat{\rho}}$ maintains a lower bound L_t on the optimum makespan, which is defined as $L_t = \max\{\frac{1}{m}p_t^+, p_t^1, 2p_t^{m+1}\}$. Here we use the same notation as in Section 3.2, introduced in Section 3.2.1. Recall that p_t^1 and p_t^{m+1} are the processing times of the largest and $(m+1)$-st largest jobs in $J_1, \ldots, J_t$, respectively. Again let $L = L_n$, and for sake of completeness we also set $L_{n+1} := L_n$.

In this section, different from before, we define a job $J_{t'}$ to be large at time t if $p_{t'} > (2\hat{\rho} - 3)L_t$; otherwise it is small at time t. Again we may say for short that a job $J_{t'}$ is large or small if it is clear from the context to which time t we refer.

Any job J_t, $1 \leq t \leq n$, is processed as follows. If J_t is small (at the time t when it is presented), then ALG$^{\hat{\rho}}$ checks if there is a machine in μ_A whose load value $\ell_s(j,t)$ is at most $(\hat{\rho} - 1)L_t$. If this is the case, then among the machines in μ_A with this property, J_t is assigned to one having the smallest $\ell_s(j,t)$ value. If there is no such machine in μ_A, then J_t is assigned to a least loaded machine in μ_B. If J_t is large, then ALG$^{\hat{\rho}}$ checks if there is machine in μ_A whose load value $\ell(j,t)$ is at most $(3-\hat{\rho})L_t$. If this is the case, then J_t is scheduled on a least loaded machine in μ_A. Otherwise J_t is assigned to a least loaded machine in μ_B.

Job Migration Phase. At any time during the phase let $\ell(j)$ denote the current load of M_j, $1 \leq j \leq m$. We first describe the job removal step. For every machine $M_j \in \mu_B$, ALG$^{\hat{\rho}}$ removes the largest job from M_j. Furthermore, while there exists a machine $M_j \in \mu_A$ whose current load exceeds $(\hat{\rho} - 1)L$, ALG$^{\hat{\rho}}$ removes the largest job from this machine M_j. Let σ_R be the set of all removed jobs. In the job reassignment step ALG$^{\hat{\rho}}$ first sorts the jobs in order of non-increasing processing times. For any i, $1 \leq i \leq |\sigma_R|$, let J_r^i be the i-th largest job in this sequence, and let p_r^i be the corresponding processing time. For $i = 1, \ldots, |\sigma_R|$, J_r^i is scheduled as follows. If there exists a machine $M_j \in \mu_B$ such that $\ell(j) + p_r^i \leq \hat{\rho}L$, i. e. J_r^i can be placed on M_j without exceeding a makespan of $\hat{\rho}L$, then J_r^i is assigned to this machine. Otherwise the job is scheduled on a least loaded machine in μ_A. A

pseudo-code description of ALG$^{\hat{\rho}}$ is given in Figure 3.2.

Theorem 3.9. *ALG$^{\hat{\rho}}$ is $\hat{\rho}$-competitive, for any constant $\hat{\rho}$ with $5/3 \leq \hat{\rho} \leq 2$.*

The proof of the above theorem is presented in Section 3.4.2.

Job arrival phase.

- Each J_t, $1 \leq t \leq n$, is scheduled as follows.
  - If J_t is small:
    - Let $\mu'_A = \{M_j \in \mu_A \mid \ell_s(j,t) \leq (\hat{\rho}-1)L_t\}$.
    - If $\mu'_A \neq \emptyset$, then assign J_t to a machine $M_j \in \mu'_A$ having the smallest $\ell_s(j,t)$ value.
    - Otherwise assign J_t to a least loaded machine $M_j \in \mu_B$.
  - If J_t is large:
    - If there is an $M_j \in \mu_A$ with $\ell(j,t) \leq (3-\hat{\rho})L_t$, then assign J_t to a least loaded machine in μ_A.
    - Otherwise assign J_t to a least loaded machine in μ_B.

Job migration phase.

- Job removal:
  - Set $\sigma_R := \emptyset$.
  - For each $M_j \in \mu_B$
    - remove the largest job J_l from M_j
    - add J_l to σ_R.
  - While there exists an $M_j \in \mu_A$ with $\ell(j) > (\hat{\rho}-1)L$
    - remove the largest job J_l from M_j
    - add J_l to σ_R.
- Job reassignment:
  - Sort the jobs of σ_R in order of non-increasing processing time.
  - For $i = 1, \ldots, |\sigma_R|$, schedule J^i_r as follows.
    - If there is an $M_j \in \mu_B$ with $\ell(j) + p^i_r \leq \hat{\rho}L$, then assign J^i_r to M_j.
    - Otherwise assign J^i_r to a least loaded machine in μ_A.

Figure 3.2: The algorithm ALG$^{\hat{\rho}}$, for $5/3 \leq \hat{\rho} \leq 2$.

In order to obtain good upper bounds on the number of job migrations, we focus on specific values of $\hat{\rho}$. First set $\hat{\rho} = 5/3$. For $\hat{\rho} = 5/3$ a job J_t is large if $p_t > 1/3 \cdot L_t$. In the arrival phase a small job is assigned to a machine in μ_A if there exists a machine in this set whose load consisting of jobs that are currently small is at most $2/3 \cdot L_t$. A large job is assigned to a machine in μ_A if there exists a machine in this set whose load is at most $4/3L_t$.

Theorem 3.10. *ALG$^{\hat{\rho}}$, with $\hat{\rho} = 5/3$, is $\frac{5}{3}$-competitive and uses at most $4m$ job migrations.*

In fact, for any $\hat{\rho}$ with $5/3 \leq \hat{\rho} \leq 2$, ALG$^{\hat{\rho}}$ uses at most $4m$ job migrations. Finally, let $\hat{\rho} = 1.75$. For $\hat{\rho} = 1.75$ a job J_t is large if $p_t > 0.5 \cdot L_t$. In the arrival phase a small job is assigned to a machine in μ_A if there is a machine in this set whose load consisting of jobs that are currently small is no more than $0.75L_t$. A large job is assigned to a machine in μ_A if there exists a machine in this set whose load is at most $1.25L_t$.

Theorem 3.11. *ALG$^{\hat{\rho}}$, with $\hat{\rho} = 1.75$, is 1.75-competitive and uses at most $2.5m$ job migrations.*

Again, for any $\hat{\rho}$ with $1.75 \leq \hat{\rho} \leq 2$, ALG$^{\hat{\rho}}$ uses at most $2.5m$ job migrations. The proofs of Theorems 3.10 and 3.11 are contained in Section 3.4.2.

3.4.2 Analysis of ALG$^{\hat{\rho}}$

In this section we analyze ALG$^{\hat{\rho}}$, for any $\hat{\rho}$ with $5/3 \leq \hat{\rho} \leq 2$, and prove Theorems 3.9, 3.10 and 3.11. We first determine the competitive ratio of ALG$^{\hat{\rho}}$ and then bound the number of job migrations performed for $\hat{\rho} = 5/3$ and $\hat{\rho} = 1.75$.

Analysis of the Competitive Ratio

We start by showing two lemmas that will allow us to bound the total load on machines in μ_B. Again, let time $n+1$ be the time when the entire job sequence $\sigma = J_1, \ldots, J_n$ has been scheduled and the migration phase starts.

Lemma 3.12. *For any time t, $1 \leq t \leq n+1$, and any $M_j \in \mu_B$, there holds $\ell(j,t) - p_l \leq (3-\hat{\rho})L_{t-1}$, where J_l with $l < t$ is the last job assigned to M_j.*

Proof. By the definition of ALG$^{\hat{\rho}}$, when J_l is assigned to M_j, all machines of μ_A have a load greater than $(\hat{\rho}-1)L_l$ and M_j is a least loaded machine in μ_B. Hence M_j's load at time l is at most $(3-\hat{\rho})L_l$ since otherwise the total load at time l on the m machines would be greater than

$$\lfloor m/2 \rfloor (\hat{\rho}-1)L_l + \lceil m/2 \rceil (3-\hat{\rho})L_l \geq mL_l \geq \sum_{i=1}^{l} p_i = p_l^+,$$

which is a contradiction. Hence

$$\ell(j,t) = \ell(j,l) + p_l \leq (3-\hat{\rho})L_l + p_l \leq (3-\hat{\rho})L_{t-1} + p_l.$$

□

Lemma 3.13. *Suppose that there exists a machine $M_{j^*} \in \mu_A$ with $\ell_s(j^*, n+1) < (2-\hat{\rho})L$. Then, for any $M_j \in \mu_B$, there holds $\ell(j, n+1) - p_l \leq (\hat{\rho}-1)L$, where J_l is the last job assigned to M_j.*

Proof. Consider any $M_j \in \mu_B$ and let J_l be the last job assigned to it. First assume that J_l is large at time l. By the definition of $\text{ALG}^{\hat{\rho}}$, at time l all machines of μ_A have a load greater than $(3-\hat{\rho})L_l$. Moreover, M_j is a least loaded machine in μ_B at time l. We argue that a least loaded machine in μ_B has a load of at most $(\hat{\rho}-1)L_l$. If this were not the case, then immediately after the assignment of J_l the total load on the m machines would be greater than

$$\begin{aligned}
&\lfloor m/2 \rfloor (3-\hat{\rho})L_l + \lceil m/2 \rceil (\hat{\rho}-1)L_l + p_l \\
&\geq (m/2 - 1/2)(3-\hat{\rho})L_l + (m/2+1/2)(\hat{\rho}-1)L_l + (2\hat{\rho}-3)L_l \\
&= mL_l + (3\hat{\rho}-5)L_l.
\end{aligned}$$

The inequality holds because $3-\hat{\rho} \geq \hat{\rho}-1$. Since $\hat{\rho} \geq 5/3$, it follows

$$\lfloor m/2 \rfloor (3-\hat{\rho})L_l + \lceil m/2 \rceil (\hat{\rho}-1)L_l + p_l \geq mL_l \geq \sum_{i=1}^{l} p_i,$$

which is a contradiction. Hence

$$\ell(j,n+1) = \ell(j,l) + p_l \leq (\hat{\rho}-1)L_l + p_l \leq (\hat{\rho}-1)L + p_l.$$

Next assume that J_l is small at time l. This implies $\ell_s(j,l) > (\hat{\rho}-1)L_l$, for all $M_j \in \mu_A$. In particular, $\ell_s(j^*,l) > (\hat{\rho}-1)L_l$. Since $\ell_s(j^*,l) \leq \ell_s(j^*,n+1) < (2-\hat{\rho})L$, it follows $L_l < (2-\hat{\rho})/(\hat{\rho}-1) \cdot L$. By Lemma 3.12, $\ell(j,l+1) \leq (3-\hat{\rho})L_l + p_l$ and we conclude

$$\begin{aligned}
\ell(j,n+1) = \ell(j,l+1) &\leq (3-\hat{\rho})L_l + p_l \\
&\leq (3-\hat{\rho})(2-\hat{\rho})/(\hat{\rho}-1) \cdot L + p_l \\
&\leq (\hat{\rho}-1)L + p_l.
\end{aligned}$$

The last inequality holds because we have $(3-\hat{\rho})(2-\hat{\rho})/(\hat{\rho}-1) \leq \hat{\rho}-1$ as $\hat{\rho} \geq 5/3$. □

We next analyze the job migration phase assuming that the job removal step has already taken place, i. e. each machine of μ_A has a load of at most $(\hat{\rho}-1)L$ and the largest job was removed from each machine of μ_B. We show that given such a machine configuration each job of σ_R can be assigned to a machine so that a load bound of $\hat{\rho}L$ is preserved. For the analysis of the reassignment step we distinguish two cases depending on whether or not at time $n+1$ all machines $M_j \in \mu_A$ have a load $\ell_s(j, n+1) \geq (2-\hat{\rho})L$. Note that in the algorithm the reassignment step is identical for both cases.

Lemma 3.14. *If $\ell_s(j, n+1) \geq (2-\hat{\rho})L$, for all $M_j \in \mu_A$, then in the reassignment step all jobs of σ_R are scheduled so that the resulting load on any of the machines is at most $\hat{\rho}L$.*

Proof. By assumption, at the end of the job arrival phase $\ell_s(j, n+1) \geq (2-\hat{\rho})L$, for all $M_j \in \mu_A$. We first show that this property is maintained throughout the job removal step. Suppose that a job J_i that is small at time $n+1$ is removed from a machine $M_j \in \mu_A$. Since ALG$^{\hat{\rho}}$ always removes the largest jobs from a machine, M_j currently contains no jobs that are large at time $n+1$. Hence M_j's current load $\ell(j)$ is equal to its current load $\ell_s(j)$ consisting of jobs that are small at time $n+1$. Since a job removal needs to be performed, $\ell_s(j) = \ell(j) > (\hat{\rho}-1)L$. Since $p_i \leq (2\hat{\rho}-3)L$, the removal of J_i leads to a load consisting of small jobs of at least $\ell_s(j) - p_l > (\hat{\rho}-1)L - (2\hat{\rho}-3)L = (2-\hat{\rho})L$.

After the job removal step each machine $M_j \in \mu_A$ has a load of at most $(\hat{\rho}-1)L$. By Lemma 3.12 each machine of μ_B has a load of at most $(3-\hat{\rho})L < \hat{\rho}L$ after ALG$^{\hat{\rho}}$ has removed the largest job from any of these machines. We show that each $J_k \in \sigma_R$ can be scheduled on a machine such that the resulting load is at most $\hat{\rho}L$. Consider any $J_k \in \sigma_R$. There holds $p_k \leq L$. Suppose that J_k cannot be feasibly scheduled on any of the machines. For any $1 \leq j \leq m$, let $\ell(j)$ denote M_j's load immediately before the assignment of J_k. If J_k cannot be placed on a machine in μ_A, then each machine $M_j \in \mu_A$ must have a load greater than $(\hat{\rho}-1)L$: If $\ell(j) \leq (\hat{\rho}-1)L$, then $\ell(j)+p_k \leq \hat{\rho}L$ and the assignment of J_k to M_j would be feasible. Hence since the start of the reassignment step each machine $M_j \in \mu_A$ must have received at least one job J_{i_j}, and the current load of M_j is $\ell(j) \geq (2-\hat{\rho})L + p_{i_j}$. Recall that the machines of μ_A are numbered $1, \ldots, \lfloor m/2 \rfloor$ and those of μ_B are numbered $\lfloor m/2 \rfloor + 1, \ldots, m$. Since J_{i_j} was not scheduled on a machine μ_B, it follows that $\ell(\lfloor m/2 \rfloor + j) + p_{i_j} > \hat{\rho}L$. Finally, since J_k cannot be placed on a machine in μ_B, we have $\ell(m) + p_k > \hat{\rho}L$.

It follows that when J_k has to be scheduled the total processing time of the jobs p_n^+ is at least

$$\sum_{j=1}^{m} \ell(j) + p_k \geq \lfloor m/2 \rfloor (2-\hat{\rho})L + \sum_{j=1}^{\lfloor m/2 \rfloor} p_{i_j} + \sum_{j=\lfloor m/2 \rfloor + 1}^{m} \ell(j) + p_k.$$

If m is even, then $\sum_{j=\lfloor m/2\rfloor+1}^{m} \ell(j) = \sum_{j=1}^{m/2} \ell(m/2+j)$. In this case we have

$$\begin{aligned}
&\sum_{j=1}^{m} \ell(j) + p_k \\
&\geq m/2 \cdot (2-\hat{\rho})L + \sum_{j=1}^{m/2} (\ell(m/2+j) + p_{i_j}) + p_k \\
&> m/2 \cdot (2-\hat{\rho})L + m/2 \cdot \hat{\rho}L \;=\; mL.
\end{aligned}$$

If m is odd, then $\sum_{j=\lfloor m/2\rfloor+1}^{m} \ell(j) = \sum_{j=1}^{\lfloor m/2\rfloor} \ell(\lfloor m/2\rfloor + j) + \ell(m)$ and

$$\begin{aligned}
\sum_{j=1}^{m} \ell(j) + p_k \;&\geq\; \lfloor m/2\rfloor \cdot (2-\hat{\rho})L + \sum_{j=1}^{\lfloor m/2\rfloor} (\ell(\lfloor m/2\rfloor + j) + p_{i_j}) + \ell(m) + p_k \\
&>\; \lfloor m/2\rfloor \cdot (2-\hat{\rho})L + \lfloor m/2\rfloor \cdot \hat{\rho}L + \hat{\rho}L \\
&=\; (m/2 - 1/2)2L + \hat{\rho}L \;>\; mL.
\end{aligned}$$

In both cases we obtain $p_n^+ \geq \sum_{j=1}^{m} \ell(j) + p_k > mL$, which contradicts the definition of L. □

Lemma 3.15. *If $\ell_s(j^*, n+1) < (2-\hat{\rho})L$, for some $M_{j^*} \in \mu_A$, then in the reassignment step all jobs of σ_R are scheduled so that the resulting load on any of the machines is at most $\hat{\rho}L$.*

Proof. In the removal step ALG$^{\hat{\rho}}$ removes the largest job from each machine $M_j \in \mu_B$. Hence, if $\ell_s(j^*, n+1) < (2-\hat{\rho})L$ for some $M_j \in \mu_A$, then by Lemma 3.13 each machine of μ_B has a load of at most $(\hat{\rho}-1)L$ after the removal step. Moreover, each machine of μ_A has a load of at most $(\hat{\rho}-1)L$ after the job removal.

Hence when the reassignment step starts, all machines have a load of at most $(\hat{\rho}-1)L$. By the definition of L each job has a processing time of at most L. Hence in the reassignment step the first m jobs can be scheduled without exceeding a load of $\hat{\rho}L$ on any of the machines. ALG$^{\hat{\rho}}$ sorts the jobs of σ_R in order of non-increasing processing times. Thus when m jobs of σ_R have been scheduled, each of the remaining jobs has a processing time of at most $1/2L$. This holds true because by the definition of L there cannot exist $m+1$ jobs of processing time greater than $1/2L$. Each job of processing time at most $1/2L$ can be scheduled on a least loaded machine without exceeding a load of $\hat{\rho}L$ since $L + 1/2L < \hat{\rho}L$. Hence every remaining job can be scheduled on a machine of μ_B or μ_A. □

Summing up, we have proven our first result in this section.

Proof. (of Theorem 3.9) Lemmas 3.14 and 3.15 imply Theorem 3.9. □

Analysis of the Job Migrations

It remains to evaluate the number of job removals in the job migration phase. We first consider $\text{ALG}^{\hat{\rho}}$ with $\hat{\rho} = 5/3$.

Lemma 3.16. *In the removal step* $\text{ALG}^{\hat{\rho}}$*, with* $\hat{\rho} = 5/3$*, removes at most seven jobs from each machine* $M_j \in \mu_A$*.*

Proof. We show that, for any $M_j \in \mu_A$, it suffices to remove at most seven jobs from M_j such that the resulting load is upper bounded by $2/3L$. The lemma then follows because in each removal operation $\text{ALG}^{\hat{\rho}}$, with $\hat{\rho} = 5/3$, removes the largest job.

First assume that $\ell_s(j, n+1) \leq 2/3L$. In this case it suffices to remove all jobs that are large at time $n+1$. Each such job has a processing time greater than $1/3L$ and was large at the time it was assigned to M_j. Consider the last time when such a job was assigned to M_j. At that time M_j had a load of at most $4/3L$ and hence could contain no more than three jobs of processing time greater than $1/3L$. Thus at time $n+1$ machine M_j contains at most four of these large jobs.

Next assume $\ell_s(j, n+1) > 2/3L$. If $\ell_s(j, n) \leq 2/3L_n$, then J_n is assigned to M_j because $L = L_n$. Hence it suffices to remove J_n and, as shown in the last paragraph, four additional jobs of processing time greater than $1/3L_n = 1/3L$.

In the following we concentrate on the case that $\ell_s(j, n+1) > 2/3L$ and $\ell_s(j, n) > 2/3L_n$. Let t^* be the earliest time such that $\ell_s(j, t) > 2/3L_t$ holds for all times $t \geq t^*$. We have $t^* > 1$ because $\ell_s(j, 1) = 0$. Thus time $t^* - 1$ is well-defined. We partition the jobs that reside on M_j at time $n+1$ into three sets. Set T_1 (set T_2) contains those jobs that were assigned to M_j at or before time $t^* - 1$ are small (large) at time $t^* - 1$. Set T_3 contains the remaining jobs, which have arrived at or after time t^*.

Claim 3.16.1. Each job of $T_2 \cup T_3$ is large at the time it is assigned to M_j.

Claim 3.16.2. There holds $\sum_{J_i \in T_1 \setminus \{J_l\}} p_i \leq 2/3L_{t^*-1}$, where J_l is the job of T_1 that was assigned last to M_j.

Claim 3.16.3. There holds $|T_2| \leq 4$.

Claim 3.16.4. For any $J_l \in T_3$, M_j's load immediately before the assignment of J_l is at most $4/3L_l$.

Claim 3.16.5. Let $J_l \in T_3$ be the last job assigned to M_j. If M_j contains at least four jobs, different from J_l, each having a processing time of at least $1/6L$, then it suffices to remove these four jobs and J_l such that M_j's resulting load is upper bounded by $2/3L$.

Claim 3.16.6. If there exists a $J_l \in T_3$ with $p_l < 1/6L$, then M_j's load immediately before the assignment of J_l is at most $2/3L$.

Claim 3.16.7. If there exists a $J_k \in T_2$ with $p_k < 1/6L$, then $\sum_{J_i \in T_1} p_i + p_k \leq 2/3L$.

Proof of Claim 3.16.1. The jobs of T_2 are large at time $t^* - 1$ and hence at the time they were assigned to M_j. By the definition of t^*, $\ell_s(j,t) > 2/3L_t$, for any $t^* \leq t \leq n$, and hence ALG$^{\hat{\rho}}$, with $\hat{\rho} = 5/3$, does not assign small jobs to M_j after time $t^* - 1$. Hence all jobs in T_3 are large when they were assigned to M_j.

Proof of Claim 3.16.2. By definition of T_1 all jobs of $T_1 \setminus \{J_l\}$ are small at time $t^* - 1$ and their total processing time is at most $\ell_s(j, t^* - 1) \leq 2/3L_{t^*-1}$ by choice of t^*.

Proof of Claim 3.16.3. Each job of T_2 has a processing time greater than $1/3L_{t^*-1}$. Consider the last time l when a job $J_l \in T_2$ was assigned to M_j. Immediately before the assignment, M_j had a load of at most $4/3L_{t^*-1}$ and hence could contain not more than three jobs of processing time greater than $1/3L_{t^*-1}$.

Proof of Claim 3.16.4. Consider any $J_l \in T_3$. By Claim 3.16.1 J_l is large at time l and hence M_j's load prior to the assignment of J_l is at most $4/3L_l$.

Proof of Claim 3.16.5. By Claim 3.16.4 M_j's load immediately before the assignment of J_l is at most $4/3L_l$. Removing four jobs of processing time at least $1/6L$ each as well as J_l reduces M_j's load to a value of at most $2/3L$.

Proof of Claim 3.16.6. By Claim 3.16.1 J_l is large at time l and hence $p_l > 1/3L_l$. Since $p_l < 1/6L$, we have $L_l < 1/2L$. By Claim 3.16.4, M_j's load immediately before the assignment of J_l is at most $4/3L_l$ and hence at most $2/3L$.

Proof of Claim 3.16.7. Job J_k is large at time $t^* - 1$ and hence $p_k > 1/3L_{t^*-1}$. Since $p_k < 1/6L$, it follows $L_{t^*-1} < 1/2L$. By Claim 3.16.2, we have $\sum_{J_i \in T_1} p_i \leq 2/3L_{t^*-1} + p_l$, where J_l is the last job of T_1 assigned to M_j. Since p_l is small at time $t^* - 1$, we have $p_l \leq 1/3L_{t^*-1} < 1/6L$. In summary $\sum_{J_i \in T_1} p_i + p_k \leq 1/3L + 1/6L + 1/6L = 2/3L$.

We proceed with the actual proof and distinguish two cases.

Case 1: If $|T_2 \cup T_3| \leq 4$, then by Claim 3.16.1 it suffices to remove the jobs of $T_2 \cup T_3$ and the last job of T_1 assigned to M_j.

Case 2: Assume $|T_2 \cup T_3| \geq 5$. Then by Claim 3.16.3 there holds $|T_2| \leq 4$ and thus $T_3 \neq \emptyset$. Let J_l be the last job of T_3 assigned to M_j. If $T_2 \cup T_3 \setminus \{J_l\}$ contains at least four jobs of processing time at least $1/6L$, then by Claim 3.16.5 it suffices to remove these four jobs and J_l. So suppose that this is not the case. Then $T_2 \cup T_3 \setminus \{J_l\}$ must contain a job of processing time smaller than $1/6L$.

Assume there exists a job in $T_3 \setminus \{J_l\}$ with this property. Then let $J_{l'}$ be the last job assigned to M_j having a processing time smaller than $1/6L$. By Claim 3.16.6, immediately before the assignment of $J_{l'}$ machine M_j has a load of at most $2/3L$.

Therefore it suffices to remove $J_{l'}$ and the jobs of T_3 subsequently scheduled on M_j. In addition to J_l, this sequence consists of at most three jobs $J_k \neq J_l$, because $T_3 \setminus \{J_l\}$ contains less than four jobs of processing time at least $1/6L$.

Finally consider the case that all jobs of $T_3 \setminus \{J_l\}$ have a processing time of at least $1/6L$ and there is a job $J_{l'} \in T_2$ having a processing time smaller than $1/6L$. By Claim 3.16.7 it suffices to remove $T_2 \setminus \{J_{l'}\} \cup T_3$. By Claim 3.16.3 we have $|T_2 \setminus \{J_{l'}\}| \leq 3$. Since $T_3 \setminus \{J_l\}$ contains less than four jobs, each having a processing time of at least $1/6L$, we have $|T_3| \leq 4$. We conclude that at most seven jobs have to be removed. □

Proof. (of Theorem 3.10). Lemma 3.13 ensures that in the job removal step $\text{ALG}^{\hat{\rho}}$, with $\hat{\rho} = 5/3$, removes at most seven jobs from any machine in μ_A. For any machine in μ_B, one job is removed. Hence the total number of migrations is at most $7\lfloor m/2 \rfloor + \lceil m/2 \rceil \leq 4m$. By Theorem 3.9 the claim on the competitive ratio follows, and this concludes the proof of Theorem 3.10. □

We next turn to the algorithm $\text{ALG}^{\hat{\rho}}$ with $\hat{\rho} = 1.75$.

Lemma 3.17. *In the job removal step* $\text{ALG}^{\hat{\rho}}$*, with* $\hat{\rho} = 1.75$*, removes at most four jobs from each machine* $M_j \in \mu_A$*.*

Proof. We show that, for any $M_j \in \mu_A$, it suffices to remove at most four jobs from M_j such that the resulting load is upper bounded by $0.75L$.

First assume that $\ell_s(j, n+1) \leq 0.75L$. Then it suffices to remove all jobs that are large at time $n+1$. Each such job has a processing time greater than $0.5L$ and was large at the time it was assigned to M_j. Consider the last time when such a job was assigned to M_j. At that time M_j had a load of at most $1.25L$ and hence could contain no more than two jobs of processing time greater than $0.5L$. Thus at time $n+1$ machine M_j contains at most three of these large jobs.

Next assume $\ell_s(j, n+1) > 0.75L$. If $\ell_s(j, n) \leq 0.75L_n$, then J_n is assigned to M_j because $L = L_n$. Hence it suffices to remove J_n and, as shown in the last paragraph, three additional jobs of processing time greater than $0.5L_n = 0.5L$.

We concentrate on the case that $\ell_s(j, n+1) > 0.75L$ and $\ell_s(j, n) > 0.75L_n$. Let t^* be the earliest time such that $\ell_s(j, t) > 0.75L_t$ holds for all times $t \geq t^*$. We partition the jobs that reside on M_j at time $n+1$ into three sets. Set T_1 (set T_2) contains the jobs that were assigned to M_j at or before time $t^* - 1$ are small (large) at time $t^* - 1$. Set T_3 contains the remaining jobs, which have arrived at or after time t^*.

Claim 3.17.1. Each job of $T_2 \cup T_3$ is large at the time it is assigned to M_j.

Claim 3.17.2. There holds $\sum_{J_i \in T_1 \setminus \{J_l\}} p_i \leq 0.75L_{t^*-1}$, where J_l is the job of T_1 that was assigned last to M_j.

Claim 3.17.3. There holds $|T_2| \leq 3$.

Claim 3.17.4. For any $J_l \in T_3$, M_j's load immediately before the assignment of J_l is at most $1.25L_l$.

Claim 3.17.5. Let $J_l \in T_3$ be the last job assigned to M_j. If M_j contains at least three jobs, different from J_l, each having a processing time of at least $1/6L$, then it suffices to remove these three jobs and J_l such that M_j's resulting load is upper bounded by $0.75L$.

Claim 3.17.6. If there exists a $J_l \in T_3$ with $p_l < 1/6L$, then M_j's load immediately after the assignment of J_l is at most $0.75L$.

Claim 3.17.7. If $T_2' \subseteq T_2$ is a subset with $1 \leq |T_2'| \leq 2$ and $p_i \leq 1/6L$, for all $J_i \in T_2'$, then $\sum_{J_i \in T_1} p_i + \sum_{J_i \in T_2'} p_i \leq 0.75L$.

Proof of Claim 3.17.1. The jobs of T_2 are large at time $t^* - 1$ and hence at the time they were assigned to M_j. By the definition of t^*, $\ell_s(j,t) > 0.75L_t$, for any $t^* \leq t \leq n$, and hence $\text{ALG}^{\hat{\rho}}$, with $\hat{\rho} = 1.75$, does not assign small jobs to M_j at times $t \geq t^*$.

Proof of Claim 3.17.2. All jobs of $T_1 \setminus \{J_l\}$ are small at time $t^* - 1$ and their total processing time is at most $\ell_s(j, t^* - 1) \leq 0.75L_{t^*-1}$, by the choice of t^*.

Proof of Claim 3.17.3. Each job of T_2 has a processing time greater than $0.5L_{t^*-1}$. Consider the last time l when a job $J_l \in T_2$ was assigned to M_j. Immediately before the assignment, M_j had a load of at most $1.25L_{t^*-1}$ and hence could contain not more than two jobs of processing time greater than $0.5L_{t^*-1}$.

Proof of Claim 3.17.4. Consider any $J_l \in T_3$. By Claim 3.17.1 J_l is large at time l and hence M_j's load prior to the assignment of J_l is at most $1.25L_l$.

Proof of Claim 3.17.5. By Claim 3.17.4 M_j's load immediately before the assignment of J_l is at most $1.25L_l$. Removing three jobs of processing time at least $1/6L$ each as well as J_l reduces M_j's load to a value of at most $0.75L$.

Proof of Claim 3.17.6. By Claim 3.17.1 J_l is large at time l and hence $p_l > 0.5L_l$. Since $p_l < 1/6L$, we have $L_l < 1/3L$. Using Claim 3.17.4 we obtain that M_j's load immediately after the assignment of J_l is at most $1.25L_l + p_l \leq 5/12L + 1/6L < 0.75L$.

Proof of Claim 3.17.7. Any job $J_i \in T_2'$ is large at time $t^* - 1$ and hence $p_i > 0.5L_{t^*-1}$. Since $p_i < 1/6L$, it follows $L_{t^*-1} < 1/3L$. By Claim 3.17.2, we have $\sum_{J_i \in T_1} p_i \leq 0.75L_{t^*-1} + p_l \leq 0.25L + 1/6L$, where J_l is the last job of T_1 assigned to M_j. Thus $\sum_{J_i \in T_1} p_i + \sum_{J_i \in T_2'} p_i \leq 0.25L + 3 \cdot 1/6L \leq 0.75L$.

We finish the proof of the lemma using a case distinction on the size of T_3.

- $|T_3| = 0$: Then by Claim 3.17.2 is suffices to remove T_2 and the last job of T_1 assigned to M_j. By Claim 3.17.3, T_2 contains no more than three jobs.

- $|T_3| = 1$: We may assume that the only job $J_l \in T_3$ has a processing time of at least $1/6L$ since otherwise by Claim 3.17.6 no job has to be removed. Moreover, we may assume that $|T_2| = 3$ since otherwise, by Claim 3.17.2 it suffices to remove $T_2 \cup T_3$ and the last job of T_1 assigned to M_j. If all the jobs of T_2 have a processing time of at least $1/6L$, then Claim 3.17.5 ensures that it suffices to remove $T_2 \cup T_3$. If one job in T_2 has a processing time of at most $1/6L$, then Claim 3.17.7 ensures that it suffices to remove the other two jobs of T_2 and T_3.

- $|T_3| = 2$: We assume that both jobs in T_3 have a processing time of at least $1/6L$ since otherwise, by Claim 3.17.6, we can just remove one job of T_3. If $|T_2| = 1$, then by Claim 3.17.2 it suffices to remove $T_2 \cup T_3$ and the last job of T_1 assigned to M_j. It remains to consider the case $|T_2| \geq 2$. If none of the jobs in T_2 has a processing time smaller than $1/6L$, then Claim 3.17.5 applies. If one of the jobs has a processing time smaller than $1/6L$, then Claim 3.17.7 applies and it suffices to remove the at most two other jobs of T_2 and the jobs of T_3.

- $|T_3| = 3$: Again we assume that all jobs in T_3 have a processing time of at least $1/6L$ since otherwise the desired statement follows from Claim 3.17.6, Moreover, we assume $|T_2| > 0$; otherwise we can apply again Claim 3.17.2. If there is one job in T_2 having a processing time of at least $1/6L$, the desired number of job removals follows from Claim 3.17.5. If this is not the case, then Claim 3.17.7 ensures that it suffices to remove the last job of T_2 assigned to M_j as well as T_3.

- $|T_3| \geq 4$: If four jobs in T_3 have a processing time of at least $1/6L$, then by Claim 3.17.5 it is sufficient to remove these four jobs. If at most three jobs have a processing time of at least $1/6L$, then let $J_l \in T_3$ be last jobs assigned to M_j having a processing time smaller than $1/6L$. By Claim 3.17.6 it suffices to remove the jobs of T_3 subsequently assigned to M_j, and there exist at most three of these.

This concludes the proof. □

Proof. (of Theorem 3.11). Recall that ALG$^{\hat{\rho}}$, with $\hat{\rho} = 1.75$, migrates $\lceil m/2 \rceil$ jobs from machines in μ_B. Hence, using the above Lemma 3.17, we obtain that the total number of migrations is at most $4\lfloor m/2 \rfloor + \lceil m/2 \rceil \leq 2.5m$. This finishes the proof of Theorem 3.11 as the claim on the competitive ratio follows by Theorem 3.9. □

We finally remark that $\Omega(m)$ migration in our algorithm ALG$^{\hat{\rho}}$ is really necessary to achieve a competitive ratio smaller than $1 + 1/\sqrt{2}$ as shown by Theorem 3.8.

3.5 Proofs of Technical Lemmas

In this section we prove the technical lemmas stated in Section 3.2 and used in Sections 3.2 and 3.3.

Proof. (of Lemma 3.1). Fix $m \geq 2$. We first evaluate $f_m(2)$ and $f_m(1+1/(3m))$. For $\rho = 2$, we have $\lceil (1-1/\rho)m \rceil \geq m/2$. Hence $\lceil (1-1/\rho)m \rceil \rho/m \geq 1$ and $f_m(2) \geq 1$. For $\rho = 1+1/(3m)$, there holds $\lceil (1-1/\rho)m \rceil = 1$. Thus $f_m(1+1/(3m)) = 1/(3m)H_{m-1} + 1/m + 1/(3m^2) < 1/3 + 1/2 + 1/12 < 1$. It remains to show that $f_m(\rho)$ is continuous and strictly increasing. To this end we show that, for any $\rho > 1$ and small $\varepsilon > 0$, $f_m(\rho+\varepsilon) - f_m(\rho)$ is strictly positive and converges to 0 as $\varepsilon \to 0$.

First consider an $\rho > 1$ such that $(1-1/\rho)m \notin \mathbb{N}$. In this case we choose $\varepsilon > 0$ such that $\lceil (1-1/(\rho+\varepsilon))m \rceil = \lceil (1-1/\rho)m \rceil$. We have

$$f_m(\rho) = (\rho-1)(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil - 1}) + \lceil (1-1/\rho)m \rceil \rho/m$$

and

$$f_m(\rho+\varepsilon) = (\rho+\varepsilon-1)(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil - 1}) + \lceil (1-1/\rho)m \rceil (\rho+\varepsilon)/m.$$

Thus

$$f_m(\rho+\varepsilon) - f_m(\rho) = \varepsilon(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil - 1}) + \lceil (1-1/\rho)m \rceil \varepsilon/m.$$

Since $\rho > 1$, there holds $\lceil (1-1/\rho)m \rceil \geq 1$, and thus $f_m(\rho+\varepsilon) - f_m(\rho) > 0$. Moreover, $f_m(\rho+\varepsilon) - f_m(\rho) \to 0$ as $\varepsilon \to 0$.

Next let $\rho > 1$ such that $(1-1/\rho)m \in \mathbb{N}$. In this case we choose $\varepsilon > 0$ such that $\lceil (1-1/(\rho+\varepsilon))m \rceil = \lceil (1-1/\rho)m \rceil + 1$. There holds

$$f_m(\rho) = (\rho-1)(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil - 1}) + \lceil (1-1/\rho)m \rceil \rho/m$$

and

$$f_m(\rho+\varepsilon) = (\rho+\varepsilon-1)(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil}) + (\lceil (1-1/\rho)m \rceil + 1)(\rho+\varepsilon)/m.$$

Taking into account that $(1-1/\rho)m \in \mathbb{N}$, we obtain

$$\begin{aligned} f_m(\rho+\varepsilon) - f_m(\rho) &= -(\rho-1) \cdot 1/((1-1/\rho)m) + \varepsilon(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil}) \\ &\quad + (\lceil (1-1/\rho)m \rceil + 1)\varepsilon/m + \rho/m \\ &= \varepsilon(H_{m-1} - H_{\lceil (1-1/\rho)m \rceil}) + (\lceil (1-1/\rho)m \rceil + 1)\varepsilon/m. \end{aligned}$$

Again, $f_m(\rho+\varepsilon) - f_m(\rho)$ is strictly positive and tends to 0 as $\varepsilon \to 0$. □

Proof. (of Lemma 3.2). We first prove that $(\rho_m)_{m\geq 2}$ is non-decreasing. A first observation is that $\rho_m \leq m$ because $f_m(m) \geq 1$. We will show that, for any $m \geq 3$ and $1 < \rho \leq m$, there holds $f_{m-1}(\rho) \geq f_m(\rho)$. This implies $1 = f_{m-1}(\rho_{m-1}) \geq f_m(\rho_{m-1})$. By Lemma 3.1, f_m is strictly increasing and thus $\rho_m \geq \rho_{m-1}$. Consider a fixed ρ with $1 < \rho \leq m$. We distinguish the cases whether or not $\lceil(1-1/\rho)(m-1)\rceil = \lceil(1-1/\rho)m\rceil$.

If $\lceil(1-1/\rho)(m-1)\rceil = \lceil(1-1/\rho)m\rceil$, then

$$\begin{aligned} f_m(\rho) &= (\rho-1)(H_{m-1} - H_{\lceil(1-1/\rho)m\rceil-1}) + \lceil(1-1/\rho)m\rceil\rho/m \\ f_{m-1}(\rho) &= (\rho-1)(H_{m-2} - H_{\lceil(1-1/\rho)m\rceil-1}) + \lceil(1-1/\rho)m\rceil\rho/(m-1). \end{aligned}$$

We obtain

$$\begin{aligned} f_{m-1}(\rho) - f_m(\rho) &= -(\rho-1)/(m-1) + \lceil(1-1/\rho)m\rceil\rho/(m(m-1)) \\ &\geq -(\rho-1)/(m-1) + (\rho-1)/(m-1) = 0, \end{aligned}$$

and thus $f_{m-1}(\rho) \geq f_m(\rho)$.

We consider the case $\lceil(1-1/\rho)(m-1)\rceil < \lceil(1-1/\rho)m\rceil$. Since $\rho > 0$, there holds $(1-1/\rho)(m-1) > (1-1/\rho)m - 1$. Thus $\lceil(1-1/\rho)m\rceil - 1 = \lceil(1-1/\rho)m-1\rceil \leq \lceil(1-1/\rho)(m-1)\rceil$. If $\lceil(1-1/\rho)(m-1)\rceil < \lceil(1-1/\rho)m\rceil$, then because of $\lceil(1-1/\rho)m\rceil - 1 \leq \lceil(1-1/\rho)(m-1)\rceil$, as just shown, it follows $\lceil(1-1/\rho)m-1\rceil = \lceil(1-1/\rho)m\rceil - 1$ and hence

$$\begin{aligned} f_m(\rho) &= (\rho-1)(H_{m-1} - H_{\lceil(1-1/\rho)m\rceil-1}) + \lceil(1-1/\rho)m\rceil\rho/m \\ f_{m-1}(\rho) &= (\rho-1)(H_{m-2} - H_{\lceil(1-1/\rho)m\rceil-2}) + (\lceil(1-1/\rho)m\rceil - 1)\rho/(m-1). \end{aligned}$$

Since $\rho > 1$, there holds $\lceil(1-1/\rho)(m-1)\rceil \geq 1$. Hence in our case $\lceil(1-1/\rho)m\rceil \geq 2$ and $\lceil(1-1/\rho)m\rceil - 1 > 0$. We obtain

$$\begin{aligned} f_{m-1}(\rho) - f_m(\rho) = &-\frac{\rho-1}{m-1} + \frac{\rho-1}{\lceil(1-1/\rho)m\rceil - 1} \\ &+ \lceil(1-1/\rho)m\rceil\frac{\rho}{m(m-1)} - \frac{\rho}{m-1}. \end{aligned}$$

Choose x, with $0 \leq x < 1$, such that $\lceil(1-1/\rho)m\rceil = (1-1/\rho)m + x$. Then

$$\begin{aligned} f_{m-1}(\rho) - f_m(\rho) &= -\frac{\rho-1}{m-1} + \frac{\rho-1}{(1-1/\rho)m + x - 1} \\ &\quad + (1-1/\rho)m\frac{\rho}{m(m-1)} + \frac{\rho x}{m(m-1)} - \frac{\rho}{m-1} \\ &= \frac{\rho-1}{(1-1/\rho)m + x - 1} + \frac{\rho x}{m(m-1)} - \frac{\rho}{m-1} \end{aligned}$$

In order to establish $f_{m-1}(\rho) - f_m(\rho) \geq 0$ it suffices to show

$$\frac{\rho - 1}{(1 - 1/\rho)m + x - 1} \geq \frac{\rho(m - x)}{m(m-1)}.$$

This is equivalent to $(\rho - 1)m(m-1) \geq (m-x)((\rho-1)m + \rho x - \rho)$. Standard algebraic manipulation yields that this is equivalent to $m \geq mx - \rho x^2 + \rho x$. Let $g(x) = mx - \rho x^2 + \rho x$, for any real number x. This function is non-decreasing for any $x \leq (m+\rho)/(2\rho)$. Since $\rho \leq m$, the function is non-decreasing for any $x \leq 1$. As $g(1) = m$, it follows that $m \geq mx - \rho x^2 + \rho x$ holds for all $0 \leq x < 1$. We conclude $f_{m-1}(\rho) - f_m(\rho) \geq 0$.

It is easy to verify that $f_2(4/3) = 1$. We show that $\lim_{m\to\infty} \rho_m$ is upper bounded by $W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2))$. Cesáro [17] proved

$$0 < H_m - \frac{1}{2}\ln\left(m(m+1)\right) - \gamma < \frac{1}{6m(m+1)}, \tag{3.1}$$

where $\gamma \approx 0.577$ is the Euler-Mascheroni constant. Using this inequality we find, for any c with $0 < c \leq 1$ and $\lceil cm \rceil - 2 > 0$,

$$\begin{aligned}
H_{m-1} - H_{\lceil cm \rceil - 2} &> \frac{1}{2}\ln((m-1)m) + \gamma - \frac{1}{2}\ln((\lceil cm \rceil - 2)(\lceil cm \rceil - 1)) \\
&\quad - \gamma - \frac{1}{6(\lceil cm \rceil - 2)(\lceil cm \rceil - 1)} \\
&\geq \frac{1}{2}\left(\ln(m-1) + \ln m - \ln(cm - 1) - \ln(cm)\right) \\
&\quad - \frac{1}{2(\lceil cm \rceil - 1)} \\
&= \frac{1}{2}\left(\ln(m-1) + \ln m - \ln(c(m - 1/c)) - \ln(cm)\right) \\
&\quad - \frac{1}{2(\lceil cm \rceil - 1)} \\
&= \frac{1}{2}\left(\ln(m-1) - \ln(m - 1/c) - 2\ln(c)\right) - \frac{1}{2(\lceil cm \rceil - 1)} \\
&\geq \frac{1}{2}\left(2\ln(1/c)\right) - \frac{1}{2(\lceil cm \rceil - 1)} \\
&\geq \ln(1/c) - \frac{1}{2(cm - 1)},
\end{aligned}$$

where the second to last inequality holds since $\ln(m - 1/c) \leq \ln(m-1)$, for $0 < c \leq 1$ and sufficiently large m. We obtain

$$\begin{aligned} f_m(\rho) &= (\rho-1)(H_{m-1} - H_{\lceil(1-1/\rho)m\rceil-1}) + (\lceil(1-1/\rho)m\rceil)\,\frac{\rho}{m} \\ &> (\rho-1)\left(\ln(\frac{\rho}{\rho-1}) - \frac{1}{2((1-1/\rho)m-1)} - \frac{1}{\lceil(1-1/\rho)m\rceil-1}\right) \\ &\quad + (\lceil(1-1/\rho)m\rceil)\,\frac{\rho}{m} \\ &\geq (\rho-1)\left(\ln(\frac{\rho}{\rho-1}) - \frac{1}{(1-1/\rho)m-1}\right) + \rho - 1 =: F(m). \end{aligned}$$

Obviously, $\lim_{m\to\infty} F(m) = (\rho-1)\ln(\frac{\rho}{\rho-1}) + \rho - 1$. We show that $(\rho - 1)\ln(\frac{\rho}{\rho-1}) + \rho - 1 = 1$, for $\rho = \frac{1}{1-\delta}$, where $\delta = -1/W_{-1}(-1/e^2)$.

Equation $(\rho-1)\ln(\frac{\rho}{\rho-1}) + \rho - 1 = 1$ is equivalent to $\ln(\frac{\rho}{\rho-1}) + 1 = \frac{1}{\rho-1}$, which in turn is equivalent to

$$\frac{\rho}{\rho-1}\cdot e = e^{\frac{1}{\rho-1}}.$$

Substituting $x = 1/(\rho-1)$, which is equivalent to $\rho = 1/x + 1$, we find that the above is equivalent to $xe + e = e^x$. Applying the Lambert W function we find that $x = -W_{-1}(-1/e^2) - 1$ is a solution of the former equality. Substituting we conclude that in fact $\rho = W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2))$ satisfies the equality. Using the same techniques we can show that $\lim_{m\to\infty}\rho_m$ is lower bounded by $W_{-1}(-1/e^2)/(1 + W_{-1}(-1/e^2))$. In the calculations, (3.1) yields that $H_{m-1} - H_{\lceil cm\rceil} < \ln(1/c) + 1/(2m)$. □

Chapter 4

Online Makespan Minimization with Parallel Schedules

In this chapter we study the problem ONLINE MINIMUM MAKESPAN SCHEDULING WITH PARALLEL SCHEDULES (MPS).

4.1 Introduction

The Model Recall that MPS is identical to classical online makespan minimization except that an algorithm is allowed to build several schedules in parallel. At the end of the scheduling process the best of these schedules is selected and the other ones are discarded. We restate the model more formally. A sequence of jobs $\sigma = J_1, \ldots, J_n$ has to be scheduled on m identical parallel machines from a set $\mu = \{M_1, \ldots, M_m\}$ so as to minimize the makespan. In the problem MPS, an algorithm ALG may maintain a set $\mathcal{S} = \{S^{(1)}, \ldots, S^{(k)}\}$ of k schedules during the scheduling process while jobs of σ arrive. Each job J_t has to be assigned in each schedule $S^{(l)}$, $1 \leq l \leq k$, to some machine. These assignments have to be done immediately and irrevocably, without knowledge of any future jobs $J_{t'}$, $t' > t$. At the end of σ, algorithm ALG selects a schedule $S^{(l)} \in \mathcal{S}$ having the smallest makespan and outputs this solution. The other schedules of $\mathcal{S}$ are deleted.

As we shall show, MPS can be reduced to the problem variant where the optimum makespan of the job sequence to be processed is known in advance. Hence let MPSO denote the variant of MPS where, prior to the arrival of the first job, an algorithm ALG is given the value of the optimum makespan $\text{OPT}(\mu, \sigma)$ for the incoming job sequence σ.

Remember that an algorithm ALG for MPS or MPSO is ρ-competitive if, for every job sequence σ, it outputs a schedule whose makespan is at most $\rho \cdot \text{OPT}(\mu, \sigma)$.

Previous Work Makespan minimization with parallel schedules was first studied by Kellerer et al. [45]. They assume that $m = 2$ machines are available and two schedules may be constructed. They show that in this case the optimal competitive ratio is equal to 4/3.

Articles [7, 9, 10, 13, 19, 45] study makespan minimization assuming that an online algorithm knows the optimum makespan or the sum of the processing times of σ. Chen et al. [19] developed a 1.6-competitive algorithm. Azar and Regev [13] showed that no online algorithm can attain a competitive ratio smaller than 4/3.

As for memory in online algorithms, Sleator and Tarjan [54] studied the paging problem assuming that an online algorithm has a larger fast memory than an offline strategy.

Our Contribution Our algorithms make use of novel guessing schemes that firstly predict the optimum makespan of a job sequence σ to within a factor of $1 + \varepsilon$ and secondly guess the job processing times and their frequencies in σ. For the latter we have to sparsify the universe of all guesses so as to reduce the number of schedules to a constant.

We present a comprehensive study of MPS. We develop a $(4/3+\varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, that uses a constant number of $1/\varepsilon^{O(\log(1/\varepsilon))}$ schedules. Furthermore, we give a $(1 + \varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, that uses a polynomial number of schedules. The number is $(m/\varepsilon)^{O(\log(1/\varepsilon)/\varepsilon)}$, which depends on m but is independent of the job sequence σ. The performance guarantees are nearly best possible. The algorithms are obtained via some intermediate results, which may be of independent interest.

First, in Section 4.2 we show that the original problem MPS can be reduced to the variant MPSO in which the optimum makespan is known. More precisely, given any ρ-competitive algorithm ALG for MPSO we construct a $(\rho + \varepsilon)$-competitive algorithm RED$^\varepsilon$, for any $0 \leq \varepsilon \leq 1$. If ALG uses k schedules, then RED$^\varepsilon$ uses $k \cdot \lceil \log(1 + \frac{6\rho}{\varepsilon}) / \log(1 + \frac{\varepsilon}{3\rho}) \rceil$ schedules. The construction works for any algorithm ALG for MPSO. In particular we could use a 1.6-competitive algorithm by Chen et al. [19] that assumes that the optimum makespan is known and builds a single schedule. We would obtain a $(1.6 + \varepsilon)$-competitive algorithm that builds at most $\lceil \log(1 + 10/\varepsilon) / \log(1 + \varepsilon/5) \rceil$ schedules.

We proceed developing algorithms for MPSO. In Section 4.3 we give a $(1+\varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, that uses $(\lfloor 2m/\varepsilon \rfloor + 1)^{\lceil \log(2/\varepsilon) / \log(1+\varepsilon/2) \rceil}$ schedules. In Section 4.4 we devise a $(4/3 + \varepsilon)$-competitive algorithm, for any $0 < \varepsilon \leq 1$, that uses $1/\varepsilon^{O(\log(1/\varepsilon))}$ schedules. Combining these algorithms with RED$^\varepsilon$, we derive the two algorithms for MPS mentioned in the above paragraph; see also Section 4.5. The number of schedules used by our strategies depends on $1/\varepsilon$ and exponentially on $\log(1/\varepsilon)$ or $1/\varepsilon$. Such a dependence seems inherent if we wish to explore the full power of parallel schedules. The trade-offs resem-

ble those exhibited by PTASes in offline approximation. Recall that the PTAS by Hochbaum and Shmoys [38] for makespan minimization achieves a $(1+\varepsilon)$-approximation with a running time of $O((n/\varepsilon)^{1/\varepsilon^2})$.

In Section 4.6 we present two lower bounds. We show that any deterministic online algorithm for MPSO that achieves a competitive ratio smaller than $1+\varepsilon$, for any $0 < \varepsilon \leq 1/3$, must construct at least $m^{\Omega(1/\varepsilon)}$ many schedules. For the specific value of $\varepsilon = 1/3$ we show a slightly better statement. We prove that any deterministic online algorithm for MPSO that achieves a competitive ratio smaller than 4/3 must construct more than $\lfloor m/3 \rfloor$ schedules. These bounds clearly transfer to MPS. Consequently, the competitive ratio of 4/3 is best possible using a constant number of schedules. Furthermore, the number of schedules of our $(1+\varepsilon)$-competitive algorithm is nearly optimal, up to a factor that is polynomial in m.

Our algorithms make use of novel guessing schemes. RED$^{\varepsilon}$ works with guesses on the optimum makespan. Guessing and *doubling* the value of the optimal solution is a technique that has been applied in other load balancing problems, see e. g. [12]. However here we have to design a refined scheme that carefully sets and readjusts guesses so that the resulting competitive ratio increases by a factor of $1+\varepsilon$ only, for any $\varepsilon > 0$. Moreover, the readjustment and job assignment rules have to ensure that scheduling errors, made when guesses were too small, are not critical. Our $(4/3+\varepsilon)$-competitive algorithm works with guesses on the job processing times and their frequencies in σ. In order to achieve a constant number of schedules, we have to sparsify the set of all possible guesses. As far as we know, such an approach has not been used in the literature before.

All our algorithms have the property that the parallel schedules are constructed basically independently. The algorithms for MPSO require no coordination at all among the schedules. In RED$^{\varepsilon}$ we only have to check whether a schedule fails, i. e. when a guess on the optimum makespan is too small. We will observe that this can be realized easily.

The competitive ratios achieved with parallel schedules are considerably smaller than the best ratios of about 1.92 known for the online scenario. We also remark that our ratio of $(4/3+\varepsilon)$, for small ε, is lower than the competitiveness of about 1.46 obtained in the semi-online settings where a reordering buffer of size $O(m)$ is available or $O(m)$ jobs may be reassigned. Skutella et al. [49] gave an online algorithm that is $(1+\varepsilon)$-competitive if, before the assignment of any job J_t, jobs of processing volume $2^{O((1/\varepsilon)\log^2(1/\varepsilon))}p_t$ may be migrated. Hence the total amount of reassignment operations used during scheduling σ depends on the length of the input sequence while in our model only a constant number of schedules is used.

4.2 Reducing MPS to MPSO

In this section we will show that any ρ-competitive algorithm ALG for MPSO can be used to construct a $(\rho+\varepsilon)$-competitive algorithm RED^{ε} for MPS, for any $\varepsilon > 0$. The main idea is to repeatedly execute ALG for a set of guesses on the optimum makespan. The initial guesses are small and are increased whenever a guess turns out to be smaller than $\text{OPT}(\mu, \sigma)$. The increments are done in small steps so that, among the final guesses, there exists one that is upper bounded by approximately $(1+\varepsilon)\text{OPT}(\mu, \sigma)$. In the analysis of this scheme we will have to bound machine loads caused by scheduling "errors" made when guesses were too small. Unfortunately the execution of RED^{ε}, given a guess $\gamma \neq \text{OPT}(\mu, \sigma)$, can lead to undefined algorithmic behavior. As we shall show in Lemma 4.1, guesses $\gamma \geq \text{OPT}(\mu, \sigma)$ are not critical. However, guesses $\gamma < \text{OPT}(\mu, \sigma)$ have to be handled carefully.

4.2.1 Preliminaries

As already mentioned in the introduction, in this chapter it will be convenient to associate schedules with algorithms, i. e. a schedule $S^{(l)}$ is maintained by an algorithm ALG_l that specifies how to assign jobs to machines in $S^{(l)}$. Thus an algorithm ALG for MPS or MPSO can be viewed as a family $\{\text{ALG}_l\}_{l\in I}$ of algorithms that maintain the various schedules, and we will write $\text{ALG} = \{\text{ALG}_l\}_{l\in I}$. If ALG is an algorithm for MPSO, then the value $\text{OPT}(\mu, \sigma)$ is of course given to all algorithms of $\{\text{ALG}_l\}_{l\in I}$. Throughout the whole chapter it will always be clear to which point in time we refer. Thus instead of $\ell(j,t)$ we use simply the notation $\ell(j)$ to denote the load of a machine M_j in some given schedule S at the considered point in time t.

Let $\text{ALG} = \{\text{ALG}_l\}_{l\in I}$ be a ρ-competitive algorithm for MPSO that, given guess γ, is executed on a job sequence σ. Upon the arrival of a job J_t, an algorithm ALG_l, with $l \in I$, may *fail* because the scheduling rules of ALG_l do not specify a machine where to place J_t in the current schedule $S^{(l)}$. We define two further conditions when an algorithm ALG_l fails. The first one identifies situations where a makespan of $\rho\gamma$ is not preserved and hence ρ-competitiveness may not be guaranteed. More precisely, ALG_l would assign J_t to a machine M_j such that $\ell(j) + p_t > \rho\gamma$, where $\ell(j)$ denotes machine M_j's load before the assignment. The second condition identifies situations where γ is not consistent with lower bounds on the optimum makespan, i. e. γ is smaller than the average machine load or the processing time of J_t. Formally, an algorithm ALG_l *fails* if a job J_t, $1 \leq t \leq n$, has to be scheduled and one of the following conditions holds.

(i) ALG_l does not specify a machine where to place J_t in the current schedule $S^{(l)}$.

(ii) There holds $\ell(j)+p_t > \rho\gamma$, for the machine M_j to which ALG_l would assign J_t in $S^{(l)}$.

(iii) There holds $\gamma < \sum_{t' \leq t} p_{t'}/m$ or $\gamma < p_t$.

We next show that guesses $\gamma \geq \text{OPT}(\mu, \sigma)$ are not problematic.

Lemma 4.1. *Let* $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$ *be a* ρ*-competitive algorithm for* MPSO *that, given guess* γ*, is executed on a job sequence* σ *with* $\gamma \geq \text{OPT}(\mu, \sigma)$*. Then there exists an algorithm* ALG_l*,* $l \in I$*, that does not fail during the processing of* σ *and generates a schedule whose makespan is at most* $\rho\gamma$*.*

Proof. Let O be an optimal schedule for the job sequence $\sigma = J_1, \ldots, J_n$. For each machine M_j in this optimal schedule with $\ell(j) < \gamma$ define a job J_{j^*} of processing time $p_{j^*} = \gamma - \ell(j)$. Let σ^* be the job sequence consisting of σ followed by the new jobs J_{j^*}. These up to m jobs may be appended to σ in any order. Obviously $\text{OPT}(\mu, \sigma^*) = \gamma$. Hence when ALG using guess γ is executed on σ^*, there must exist an algorithm ALG_{l^*}, $l^* \in I$, that generates a schedule with a makespan of at most $\rho\gamma$. Since σ is a prefix of σ^*, this algorithm ALG_{l^*} does not fail and generates a schedule with a makespan of at most $\rho\gamma$, when ALG given guess γ is executed on σ. □

4.2.2 Description of the Reduction

We describe an algorithm $\text{ALG}^{\varepsilon,h}$ for MPS, where $\varepsilon > 0$ and $h \in \mathbb{N}$ may be chosen arbitrarily. The construction takes as input any algorithm $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$ for MPSO. For a proper choice of h, $\text{ALG}^{\varepsilon,h}$ will be $(\rho + \varepsilon)$-competitive, provided that ALG is ρ-competitive. For this specific choice of h algorithm $\text{ALG}^{\varepsilon,h}$ is the mentioned algorithm RED^{ε}.

At any time $\text{ALG}^{\varepsilon,h}$ works with h guesses $\gamma_1 < \ldots < \gamma_h$ on the optimum makespan for the incoming job sequence σ. These guesses may be adjusted during the processing of σ; the update procedure will be described in detail below. For each guess γ_i, $1 \leq i \leq h$, $\text{ALG}^{\varepsilon,h}$ executes ALG. Hence $\text{ALG}^{\varepsilon,h}$ maintains a total of $h|I|$ schedules, which can be partitioned into subsets $\mathcal{S}_1, \ldots, \mathcal{S}_h$. Subset $\mathcal{S}_i$ contains those schedules generated by ALG using γ_i, $1 \leq i \leq h$. Let $S^{(il)} \in \mathcal{S}_i$ denote the schedule generated by ALG_l using γ_i.

A job sequence σ is processed as follows. Initially, upon the arrival of the first job J_1, the guesses are initialized as $\gamma_1 = p_1$ and $\gamma_i = (1+\varepsilon)\gamma_{i-1}$, for $i = 2, \ldots, h$. Each job J_t, $1 \leq t \leq n$, is sequenced in every schedule $S^{(il)}$, $1 \leq i \leq h$ and $1 \leq l \leq |I|$ as follows. Algorithm $\text{ALG}^{\varepsilon,h}$ checks if ALG_l using γ_i fails when having to sequence J_t in $S^{(il)}$. We remark that this check can be performed easily by just verifying if one of the conditions (i)–(iii) holds. If ALG_l using γ_i does not

fail and has not failed since the last adjustment of γ_i, then in $S^{(il)}$ job J_t is assigned to the machine specified by ALG_l using γ_i. Here the initialization of a guess is also regarded as an adjustment. If ALG_l using γ_i does fail, then J_t and all future jobs are always assigned to a least loaded machine in $S^{(il)}$ until γ_i is adjusted the next time.

Suppose that after the sequencing of J_t all algorithms of $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$ using a particular guess γ_i have failed since the last adjustment of this guess. Let i^* be the largest index i with this property. Then the guesses $\gamma_1, \ldots, \gamma_{i^*}$ are adjusted. Set $\gamma_1 = (1+\varepsilon)\max\{\gamma_h, p_t, \sum_{1 \le t' \le t} p_{t'}/m\}$ and $\gamma_i = (1+\varepsilon)\gamma_{i-1}$, for $i = 2, \ldots, i^*$. For any readjusted guess γ_i, $1 \le i \le i^*$, algorithm ALG using γ_i ignores all jobs $J_{t'}$ with $t' < t$ when processing future jobs of σ. More precisely, when making scheduling decisions and determining machine loads, algorithm ALG_l using γ_i ignores all jobs $J_{t'}$ with $t' < t$ in its schedule $S^{(il)}$. These jobs are also ignored when $\text{ALG}^{\varepsilon,h}$ checks if ALG_l using guess γ_i fails on the arrival of a job. Furthermore, after the assignment of J_t, the machines in $S^{(il)}$ are renumbered so that J_t is located on such a machine that it would occupy if it were the first job of an input sequence.

When guesses have been adjusted, they are renumbered as well as the corresponding schedule sets $\mathcal{S}_i$, such that again $\gamma_1 < \ldots < \gamma_h$. Hence after the renumbering we have again that $S^{(il)}$ is the schedule maintained by algorithm ALG_l using guess γ_i and, moreover, $\gamma_1 = \min_{1 \le i \le h} \gamma_i$ and $\gamma_i \ge (1+\varepsilon)\gamma_{i-1}$, for $i = 2, \ldots, h$. We also observe that whenever a guess is adjusted, its value increases by a factor of at least $(1+\varepsilon)^h$. A summary of $\text{ALG}^{\varepsilon,h}$ is given in Figure 4.1.

We obtain the following result, which is proved in the next subsection.

Theorem 4.2. *Let* $\text{ALG} = \{\text{ALG}_l\}_{l \in I}$ *be a* ρ*-competitive algorithm for* MPSO. *Then for any* $0 < \varepsilon \le 1$ *and* $h = \lceil \log(1 + \frac{6\rho}{\varepsilon}) / \log(1 + \frac{\varepsilon}{3\rho}) \rceil$, *algorithm* $\text{RED}^\varepsilon = \text{ALG}^{\varepsilon/(3\rho),h}$ *is* $(\rho+\varepsilon)$*-competitive for* MPS. RED^ε *uses* $h \cdot |I|$ *many schedules.*

4.2.3 Analysis of the Algorithm

In this section we analyze RED^ε. The next lemma upper bounds the value of the smallest guess when $\text{ALG}^{\varepsilon,h}$ has finished processing a job sequence σ.

Lemma 4.3. *After* $\text{ALG}^{\varepsilon,h}$ *has processed a job sequence* σ*, there holds* $\gamma_1 \le (1+\varepsilon)\text{OPT}(\mu, \sigma)$.

Proof. At any time $\text{ALG}^{\varepsilon,h}$ maintains h guesses. We can view these guesses as being stored in h variables. A variable is updated whenever its current guess is increased. Hence during the processing of σ a variable may take any position in the sorted sequence of guesses. We analyze the steps in which $\text{ALG}^{\varepsilon,h}$ adjusts guesses.

1. Set $\gamma_i = p_1(1+\varepsilon)^{i-1}$, for $i = 1, \ldots, h$.
2. For each incoming job J_t execute the following steps.
   (a) J_t is sequenced as follows in each $S^{(il)}$.
      (i) If ALG_l using γ_i fails or has failed since the last adjustment of γ_i on its schedule $S^{(il)}$, then assign J_t to a least loaded machine in $S^{(il)}$.
      (ii) Otherwise assign J_t in $S^{(il)}$ to the machine specified by ALG_l, ignoring jobs that arrived before the last adjustment of γ_i.
   (b) If all algorithms $\{\text{ALG}_l\}_{l\in I}$ for some γ_i have failed on their respective schedules $S^{(il)}$ since the last readjustment of γ_i, then let i^* be the largest index of such a γ_i, and execute the following steps.
      (i) Set $\gamma_i = (1+\varepsilon)^i \max\{\gamma_h, p_t, \sum_{t'\le t} p_{t'}/m\}$, for $i = 1, \ldots, i^*$.
      (ii) Renumber the guesses such that $\gamma_1 < \ldots < \gamma_h$.
      (iii) Renumber the schedules such that ALG_l using guess γ_i maintains schedule $S^{(il)}$.

Figure 4.1: The algorithm $\text{ALG}^{\varepsilon,h}$.

We first show that when $\text{ALG}^{\varepsilon,h}$ adjusts a guess γ, then $\gamma < \text{OPT}(\mu,\sigma)$. So suppose that after the arrival of a job J_t, $\text{ALG}^{\varepsilon,h}$ adjusts guesses $\gamma_1, \ldots, \gamma_{i^*}$, where i^* is the largest index i such that all algorithms $\{\text{ALG}_l\}_{l\in I}$ using γ_i have failed. We prove $\gamma_{i^*} < \text{OPT}(\mu,\sigma)$, which implies the desired statement because guesses are numbered in order of increasing value. Let t^*, with $t^* < t$, be the most recent time when the variable storing γ_{i^*} was updated last. If the variable has never been updated since its initialization, then let $t^* = 1$. All the algorithms $\{\text{ALG}_l\}_{l\in I}$ using γ_{i^*} ignore the jobs having arrived before J_{t^*} when making scheduling decisions for $J_{t^*}, \ldots, J_t$. Let $\sigma^* = J_{t^*}, \ldots, J_t$. There holds $\text{OPT}(\mu,\sigma^*) \le \text{OPT}(\mu,\sigma)$. If $\gamma_{i^*} \ge \text{OPT}(\mu,\sigma)$ held true, then by Lemma 4.1 there would be an algorithm ALG_{l^*}, $l^* \in I$, that, using guess γ_{i^*}, does not fail when handling σ^*. This contradicts the fact that at time t all algorithms $\{\text{ALG}_l\}_{l\in I}$ using γ_{i^*} fail or have failed since the arrival of J_{t^*}.

Let γ_1^e denote the value of the smallest guess when $\text{ALG}^{\varepsilon,h}$ has finished processing σ. We distinguish two cases depending on whether or not the variable storing γ_1^e has ever been updated since its initialization. If the variable has never been updated, then $\gamma_1^e = p_1(1+\varepsilon)^{i-1}$, for some $i \in \{1, \ldots, h\}$. If $i = 1$, there is nothing to show because $p_1 \le \text{OPT}(\mu,\sigma)$. If $i > 1$, then the initial guess of value $\gamma_{i-1} = p_1(1+\varepsilon)^{i-2}$ must have been adjusted. This implies, as shown above,

$\gamma_{i-1} < \text{OPT}(\mu, \sigma)$ and the lemma follows because $\gamma_1^e = (1+\varepsilon)\gamma_{i-1}$.

In the remainder of the proof we assume that the variable g storing γ_1^e has been updated. Consider the last update of g before the end of σ and suppose that it took place on the arrival of job J_{t^*}. First assume that g stores the smallest guess, among the h guesses, before the update. Then $\gamma_1^e = (1+\varepsilon)\max\{\gamma^*, p_{t^*}, \sum_{1\leq t'\leq t^*} p_{t'}/m\}$, where γ^* is the largest guess before the update. If γ^* is also adjusted on the arrival of J_{t^*}, then we are done because, as shown above, $\gamma^* < \text{OPT}(\mu, \sigma)$ and thus $\max\{\gamma^*, p_{t^*}, \sum_{1\leq t'\leq t^*} p_{t'}/m\} \leq \text{OPT}(\mu, \sigma)$. If γ^* is not adjusted on the arrival of J_{t^*}, then γ_1^e is the smallest guess greater than γ^* after the update. By the end of σ guess γ^* must be adjusted since otherwise γ_1^e cannot become the smallest guess. Again $\gamma^* < \text{OPT}(\mu, \sigma)$ and we are done.

Finally assume that before the update g does not store the smallest guess. Let g' be the variable that stores the largest guess smaller than that in g. After the update there holds $\gamma_1^e = (1+\varepsilon)\gamma$, where γ is the guess stored in g' after the update. Until the end of σ, γ must be adjusted again since otherwise γ_1^e cannot become the smallest guess. Again $\gamma < \text{OPT}(\mu, \sigma)$ and hence $\gamma_1^e < (1+\varepsilon)\text{OPT}(\mu, \sigma)$. □

We can now prove Theorem 4.2.

Proof. (of Theorem 4.2). Throughout the proof let $h = \lceil \log(1+\frac{6\rho}{\varepsilon}) / \log(1+\frac{\varepsilon}{3\rho}) \rceil$ and $\text{RED}^\varepsilon = \text{ALG}^{\varepsilon/(3\rho),h}$. Consider an arbitrary job sequence and let γ_1 be the smallest of the h guesses maintained by RED^ε at the end of σ. Let $\mathcal{S}_1$ be the set of schedules associated with γ_1, i. e. $\mathcal{S}_1$ was generated by $\text{ALG} = \{\text{ALG}_l\}_{l\in I}$ using a series of guesses ending with γ_1. Let $\gamma(0) < \ldots < \gamma(s)$, with $s \geq 0$, be this series and g be the variable that stored these guesses. Here $\gamma(0)$ is one of the initial guesses and $\gamma(s) = \gamma_1$.

A first observation is that at the end of σ there exists an algorithm ALG_{l^*}, $l^* \in I$, that using γ_1 has not failed. This holds true in case that g was set to $\gamma_1 = \gamma(s)$ upon the arrival of a job J_t with $t < n$ because the failure of all algorithms $\{\text{ALG}_l\}_{l\in I}$ using γ_1 would have caused an adjustment of γ_1. This also holds true if g was set to γ_1 upon the arrival of J_n because in this case none of the algorithms $\{\text{ALG}_l\}_{l\in I}$ using γ_1 has failed at the end of σ. So let ALG_{l^*}, $l^* \in I$, be an algorithm that using γ_1 has not failed and let $S^{(1l^*)}$ be the associated schedule. We prove that the load of every machine in $S^{(1l^*)}$ is upper bounded by $(\rho+\varepsilon)\text{OPT}(\mu, \sigma)$. This establishes the theorem.

Let $t_0 = 1$. If the variable g was updated during the processing of σ, then let $t_1, \ldots, t_s$ be these points in time, i. e. the arrival of J_{t_i} caused an update of g and the variable was set to $\gamma(i)$, $1 \leq i \leq s$. For any machine M_j, $1 \leq j \leq m$, in $S^{(1l^*)}$ let $\ell(j)$ denote its final load at the end of σ. Moreover, let $\ell_{t_i}(j)$ denote its load

due to jobs J_t with $t \geq t_i$, for $i = 0, \ldots, s$. Obviously

$$\ell(j) = \ell_{t_s}(j) + \sum_{i=0}^{s-1} \left(\ell_{t_i}(j) - \ell_{t_{i+1}}(j)\right). \tag{4.1}$$

We first show that $\ell_{t_s}(j) \leq \rho\gamma_1$. Immediately after J_{t_s} has been scheduled, M_j's load consisting of jobs $J_{t'}$ with $t' \geq t_s$ is at most p_{t_s}. Since g was set to $\gamma(s) = \gamma_1$ on the arrival of J_{t_s}, the guess adjustment rule ensures $p_{t_s} \leq \gamma_1$. Until the end of σ algorithm A_{k^*} using γ_1 does not fail and hence condition (ii) specifying the failure of algorithms implies that the assignment of each further job does not create a machine load greater than $\rho\gamma_1$ in $S^{(1l^*)}$.

We next show $\ell_{t_i}(j) - \ell_{t_{i+1}}(j) \leq \max\{\rho, 2\}\gamma(i)$, for each $i = 0, \ldots, s-1$. The latter difference is the load on machine M_j caused by jobs in the set $J_{t_i}, \ldots, J_{t_{i+1}-1}$. Hence it suffices to show that after the assignment of any J_t, with $t_i \leq t < t_{i+1}$, M_j's load due to jobs $J_{t'}$, with $t' \geq t_i$, is at most $\max\{\rho, 2\}\gamma(i)$. After the assignment of J_{t_i} M_j's respective load is at most p_{t_i} and this value is upper bounded by $\gamma(i)$ as ensured by the guess adjustment rule. At times $t > t_i$, while ALG_{l^*} using $\gamma(i)$ has not failed, M_j's load due to jobs $J_{t'}$ with $t' \geq t_i$ does not exceed $\rho\gamma(i)$ as ensured by condition (ii) specifying the failure of algorithms. Finally consider a time t, $t_i < t < t_{i+1}$, at which ALG_{l^*} fails or has failed. The incoming job J_t is assigned to a least loaded machine. Hence if J_t is placed on M_j, then the resulting machine load due to jobs $J_{t'}$ with $t' \geq t_i$ is upper bounded by $\sum_{t_i \leq t' < t} p_{t'}/m + p_t \leq \sum_{1 \leq t' \leq t} p_{t'}/m + p_t$. Observe that after the arrival of J_t there exists an algorithm ALG_l, $l \in I$, that using $\gamma(i)$ has not yet failed, since otherwise $\gamma(i)$ would be adjusted before time t_{i+1}. Condition (iii) defining the failure of algorithms ensures that $\sum_{1 \leq t' \leq t} p_{t'}/m \leq \gamma(i)$ and $p_t \leq \gamma(i)$. We obtain that M_j's machine load is at most $2\gamma(i)$.

We conclude that (4.1) is upper bounded by

$$\rho\gamma_1 + \sum_{i=0}^{s-1} \max\{\rho, 2\}\gamma(i). \tag{4.2}$$

By Lemma 4.3, $\gamma_1 = \gamma(s) \leq (1+\varepsilon/(3\rho))\text{OPT}(\mu, \sigma)$. At the end of the description of $\text{ALG}^{\varepsilon,h}$ we observed that whenever a guess is adjusted, it is increased by a factor of at least $(1+\varepsilon)^h$. Hence $\gamma(i) \geq (1+\varepsilon/(3\rho))^h\gamma(i-1)$. It follows that

$$\gamma(i) \leq \frac{\gamma(s)}{(1+(\varepsilon/3\rho))^{(s-i)\cdot h}},$$

for every $0 \leq i \leq s$. Hence (4.2) is upper bounded by

$$\rho(1+\frac{\varepsilon}{3\rho})\text{OPT}(\mu, \sigma) + \sum_{i=0}^{s-1} \frac{\max\{\rho, 2\}\gamma(s)}{(1+\varepsilon/(3\rho))^{h\cdot(s-i)}}$$

$$\leq \rho(1+\frac{\varepsilon}{3\rho})\text{OPT}(\mu,\sigma) + \rho(1+\frac{\varepsilon}{3\rho})\text{OPT}(\mu,\sigma)\sum_{i=0}^{s-1}\frac{2}{(1+\varepsilon/(3\rho))^{h\cdot(s-i)}} \quad (4.3)$$

$$\leq \rho(1+\frac{\varepsilon}{3\rho})\text{OPT}(\mu,\sigma)\left(1+\sum_{i=1}^{\infty}\frac{2}{(1+\varepsilon/(3\rho))^{h\cdot i}}\right)$$

$$= \rho(1+\frac{\varepsilon}{3\rho})\text{OPT}(\mu,\sigma)\left(1+\frac{2}{(1+\varepsilon/(3\rho))^{h}-1}\right) \quad (4.4)$$

$$\leq \rho(1+\frac{\varepsilon}{3\rho})^2\text{OPT}(\mu,\sigma) \;\leq\; \rho(1+\frac{\varepsilon}{\rho})\text{OPT}(\mu,\sigma) = (\rho+\varepsilon)\text{OPT}(\mu,\sigma). \quad (4.5)$$

In (4.3) we use the fact that $\max\{\rho,2\} \leq 2\rho$ and apply Lemma 4.3. Line (4.4) follows from the Geometric Series. Finally, Line (4.5) is by the choice of h and because $0 < \varepsilon \leq 1$. □

4.3 A $(1+\varepsilon)$-competitive Algorithm for MPSO

We present an algorithm PTAS$^{\varepsilon}$ for MPSO that attains a competitive ratio of $1+\varepsilon$, for any $\varepsilon > 0$. The number of parallel schedules used by PTAS$^{\varepsilon}$ will be $(\lfloor 2m/\varepsilon \rfloor + 1)^{\lceil \log(2/\varepsilon)/\log(1+\varepsilon/2)\rceil}$. This is a polynomial in m though not constant. Nonetheless it is close to being best possible since we can show in Theorem 4.17 that a polynomial number of schedules in m is necessary to achieve a competitiveness smaller than $1+\varepsilon$, and the degree of this polynomial depends linear on $1/\varepsilon$. Furthermore, PTAS$^{\varepsilon}$ will also be useful in the next section where we develop a $(4/3+\varepsilon)$-competitive algorithm for MPSO. There PTAS$^{\varepsilon}$ will be used as subroutine for a small, constant number of m.

Description of PTAS$^{\varepsilon}$. Let $\varepsilon > 0$ be arbitrary. Recall that in MPSO the optimum makespan $\text{OPT}(\mu,\sigma)$ for the incoming job sequence σ is initially known. Assume w. l. o. g. that $\text{OPT}(\mu,\sigma) = 1$. Then all job processing times are in $(0,1]$. Set $\varepsilon' = \varepsilon/2$. First we partition the range of possible job processing times into intervals $I_0,\ldots,I_k$ such that, within each interval I_i with $i \geq 1$, the values differ by a factor of at most $1+\varepsilon'$. Such a partitioning is standard and has been used e. g. in the PTAS for offline makespan minimization devised by Hochbaum and Shmoys [38]. Let $k = \lceil \log(1/\varepsilon')/\log(1+\varepsilon')\rceil$. Set $I_0 = (0,\varepsilon']$ and $I_i = ((1+\varepsilon')^{i-1}\varepsilon', (1+\varepsilon')^{i}\varepsilon']$, for $i = 1,\ldots,k$. Obviously $I_0 \cup \ldots \cup I_k = (0,(1+\varepsilon')^{k}\varepsilon']$ and $(0,1] \subseteq (0,(1+\varepsilon')^{k}\varepsilon']$. In this section we call a job *small* if its processing time is at most ε' and hence contained in I_0; otherwise we call the job *large*.

Each job sequence σ with $\text{OPT}(\mu, \sigma) = 1$ contains at most $\lfloor m/\varepsilon' \rfloor$ large jobs. For each possible sequence of large jobs algorithm $\text{PTAS}^{\varepsilon}$ uses one algorithm ALG_v or, equivalently, maintains one schedule $S^{(v)}$.

Let $V = \{(v_1, \ldots, v_k) \in \mathbb{N}_0^k \mid v_i \leq \lfloor m/\varepsilon' \rfloor\}$. There holds $|V| = (\lfloor m/\varepsilon' \rfloor + 1)^k$. Let $\text{PTAS}^{\varepsilon} = \{\text{ALG}_v\}_{v \in V}$. For any vector $v = (v_1, \ldots, v_k) \in V$, algorithm ALG_v works as follows. It assumes that the incoming job sequence σ contains exactly v_i jobs with a processing time in I_i, for $i = 1, \ldots, k$. Moreover, it pessimistically assumes that each processing time in I_i takes the largest possible value $(1+\varepsilon')^i\varepsilon'$. Hence, initially ALG_v computes an optimal schedule $O^{(v)}$ for a job sequence consisting of v_i jobs with a processing time of $(1+\varepsilon')^i\varepsilon'$, for $i = 1, \ldots, k$. Small jobs are ignored. Since running time is not an issue in the design of online algorithms, such a schedule $O^{(v)}$ can be computed exactly. Alternatively, an $(1+\varepsilon')$-approximation to the optimal schedule can be computed using the PTAS by Hochbaum and Shmoys [38]. For any (partial) schedule $S = (S_1, \ldots, S_m)$ let $n_i(S_j)$ denote the number of jobs with a processing time in I_i assigned to machine M_j in S, which is formally $n_i(S_j) = |\{J_t \in S_j \mid p_t \in I_i\}|$, where $1 \leq i \leq k$ and $1 \leq j \leq m$.

Note that for schedule $O^{(v)}$ the value $n_i(O_j^{(v)})$ counts the number of jobs whose processing time equals $(1+\varepsilon')^i\varepsilon'$, by the input sequence for which $O^{(v)}$ was constructed. Also notice that for any partial schedule $S^{(v)} = (S_1^{(v)}, \ldots, S_m^{(v)})$, value $n_i(S_j^{(v)})$ counts the number of jobs with a processing time in I_i that have been assigned to machine M_j in $S^{(v)}$ so far.

When processing the actual job sequence σ and constructing a real schedule $S^{(v)}$, ALG_v uses $O^{(v)}$ as a guideline to make scheduling decisions though the job sequences for which these schedules $S^{(v)}$ and $O^{(v)}$ are built may differ. Each incoming job J_t, $1 \leq t \leq n$, is handled as follows. If J_t is large, then let I_i with $1 \leq i \leq k$ be the interval such that $p_t \in I_i$. Algorithm ALG_v checks if there is a machine M_j such that $n_i(O_j^{(v)}) - n_i(S_j^{(v)}) > 0$, i. e. whether there is a machine that can still accept a job with a processing time in I_i as suggested by the optimal schedule $O^{(v)}$. If such a machine M_j exists, then J_t is assigned to it; otherwise J_t is scheduled on an arbitrary machine. If J_t is small, then J_t is assigned to a machine M_j with the smallest current value $\ell(O_j^{(v)}) + \ell_s(S_j^{(v)})$. Here $\ell_s(S_j^{(v)})$ denotes the current load on machine M_j caused by small jobs in $S^{(v)}$, i. e. for jobs with processing time in I_0. A summary of $\text{PTAS}^{\varepsilon}$ is given in Figure 4.2.

Theorem 4.4. *For any $\varepsilon > 0$, $\text{PTAS}^{\varepsilon}$ is $(1+\varepsilon)$-competitive and uses at most $(\lfloor 2m/\varepsilon \rfloor + 1)^{\lceil \log(2/\varepsilon)/\log(1+\varepsilon/2) \rceil}$ schedules.*

Proof. The bound on the number of schedules simply follows from the fact that $\text{PTAS}^{\varepsilon}$ maintains $|V| = (\lfloor m/\varepsilon' \rfloor + 1)^k$ schedules where again $\varepsilon' = \varepsilon/2$ and $k = \lceil \log(1/\varepsilon')/\log(1+\varepsilon') \rceil$.

1. Enumerate $V = \{(v_1, \ldots, v_k) \in \mathbb{N}_0^k \mid v_i \leq \lfloor m/\varepsilon' \rfloor\}$, with $\varepsilon' = \varepsilon/2$ and $k = \lceil \log(1/\varepsilon')/\log(1+\varepsilon') \rceil$.
2. For each $v \in V$, with $v = (v_1, \ldots, v_k)$, compute an optimal schedule $O^{(v)}$ for the input sequence consisting of v_i jobs having processing time $(1+\varepsilon')^i\varepsilon'$, $1 \leq i \leq k$.
3. Sequence each incoming job J_t into every $S^{(v)}$ using ALG_v in Figure 4.3, where ALG_v is given access to $O^{(v)}$.

Figure 4.2: The algorithm PTAS^ε.

1. Let J_t be the job that has to be assigned.
2. If $p_t > \varepsilon'$, then assign J_t as follows.
   (a) Determine I_i such that $p_t \in I_i$.
   (b) If $\exists\, M_j$ with $n_i(O_j^{(v)}) - n_i(S_j^{(v)}) > 0$, then assign J_t to M_j.
   (c) Otherwise assign J_t to an arbitrary machine.
3. If $p_t \leq \varepsilon'$, then assign J_t to a machine M_j with the smallest value $\ell(O_j^{(v)}) + \ell_s(S_j^{(v)})$.

Figure 4.3: The algorithm ALG_v.

Let σ be an arbitrary job sequence and let v_i be the number of jobs in σ with a processing time in I_i, for $i = 1, \ldots, k$. Since any v_i is upper bounded by $\lfloor m/\varepsilon' \rfloor$, the resulting vector $v = (v_1, \ldots, v_k)$ is in V. For this vector v, consider the associated algorithm ALG_v. We prove that when ALG_v has finished processing σ, the resulting schedule $S^{(v)}$ has a makespan of at most $1 + \varepsilon = (1+\varepsilon)\text{OPT}(\mu, \sigma)$.

We analyze the steps in which ALG_v assigns jobs J_t, $1 \leq t \leq n$, to machines in $S^{(v)}$. If J_t is large with $p_t \in I_i$, $1 \leq i \leq k$, then there must exist a machine M_j in the current schedule $S^{(v)}$ such that $n_i(O_j^{(v)}) - n_i(S_j^{(v)}) > 0$. Algorithm ALG_v will assign J_t to such a machine. Hence after the processing of σ, for any M_j in $S^{(v)}$, the total load caused by large jobs is upper bounded by $\ell(O_j^{(v)})$. We next argue that this value is at most $(1+\varepsilon')\text{OPT}(\mu, \sigma)$. Consider an optimal schedule O for σ. Modify this schedule by deleting all small jobs and rounding each job processing time in I_i to $(1+\varepsilon')^i\varepsilon'$, for $i = 1, \ldots, k$. The resulting schedule O' has a makespan of at most $(1+\varepsilon')\text{OPT}(\mu, \sigma)$. Furthermore O' is a schedule for an input sequence consisting of v_i jobs of processing time $(1+\varepsilon')^i\varepsilon'$. Since $O^{(v)}$

is an optimal schedule for this input, each machine load $\ell(O_j^{(v)})$ is upper bounded by $(1+\varepsilon')\text{OPT}(\mu,\sigma)$.

We finally show that when ALG_v has to sequence a small job J_t, then there is a machine M_j such that $\ell(O_j^{(v)}) + \ell_s(S_j^{(v)})$ is upper bounded by $(1+\varepsilon')\text{OPT}(\mu,\sigma)$. This implies that the assignment of J_t causes a machine load of at most $(1+\varepsilon')\text{OPT}(\mu,\sigma) + p_t \leq (1+2\varepsilon')\text{OPT}(\mu,\sigma) = (1+\varepsilon)\text{OPT}(\mu,\sigma)$ in the final schedule $S^{(v)}$.

So suppose that upon the arrival of a small job J_t there holds $\ell(O_j^{(v)}) + \ell_s(S_j^{(v)}) > (1+\varepsilon')\text{OPT}(\mu,\sigma)$ for all machines M_j, $1 \leq j \leq m$. Recall that $\ell_s(S_j^{(v)})$ is the load on machine M_j caused by small jobs in the current schedule $S^{(v)}$. Note that $\sum_{j=1}^m \ell(O_j^{(v)})$ is the total processing time of large jobs in σ if processing times in I_i are rounded up to $(1+\varepsilon')^i\varepsilon'$, for $i = 1,\ldots,k$. Hence $1/(1+\varepsilon)\sum_{j=1}^m \ell(O_j^{(v)})$ is a lower bound on the total processing time of the large jobs in σ. It follows that the total processing time of all jobs in σ is at least $1/(1+\varepsilon')\sum_{j=1}^m \ell(O_j^{(v)}) + \sum_{j=1}^m \ell_s(S_j^{(v)}) + p_t \geq 1/(1+\varepsilon')\sum_{j=1}^m (\ell(O_j^{(v)}) + \ell_s(S_j^{(v)})) + p_t$. The assumption that $\ell(O_j^{(v)}) + \ell_s(S_j^{(v)}) > (1+\varepsilon')\text{OPT}(\mu,\sigma)$ holds for all machines M_j implies that the total processing time of jobs in σ is at least $m\cdot\text{OPT}(\mu,\sigma) + p_t > m\cdot\text{OPT}(\mu,\sigma)$, which contradicts the fact that $\text{OPT}(\mu,\sigma)$ is the optimum makespan. □

4.4 A $(4/3+\varepsilon)$-competitive Algorithm for MPSO

In this section we develop an algorithm for MPSO that is $(4/3+\varepsilon)$-competitive, for any $0 < \varepsilon \leq 1$. The number of required schedules is $1/\varepsilon^{O(\log(1/\varepsilon))}$, which is a constant independent of n and m. We first present the algorithm and then analyze it.

4.4.1 Description of the Algorithm

We develop an algorithm ALG^ε that is $(4/3+\varepsilon)$-competitive, for any $0 < \varepsilon \leq 1$, if the number m of machines is not too small. We then combine ALG^ε with PTAS^ε, presented in the last section, and derive a strategy $\text{ALG}^{\hat{\varepsilon}}$ that is $(4/3+\varepsilon)$-competitive, for arbitrary m.

Before describing ALG^ε in detail, we explain the main ideas of the algorithm. One concept is similar to that used by PTAS^ε: We partition the range of possible job processing times into intervals or *job classes*. However, in order to achieve that only a substantially smaller number of schedules is needed, we have to choose the job classes more carefully. Additionally, they have to be chosen in such a way that a compact packing of jobs on the machines is possible.

An important aspect in the construction of ALG^ε is that not all vectors $v \in V$ are considered, where V contains similarly as above all vectors describing the number of large jobs of the respective job classes in the input sequence. Instead we will define a suitable sparsification V' of V. Each $v \in V'$ represents an estimate or guess on the number of large jobs arising in σ. More precisely, if $v = (v_1, \ldots, v_k)$, then it is assumed that σ contains at least v_i jobs with a processing time of job class i.

Obviously, the job sequence σ may contain more large jobs, and their exact number is unknown. As a consequence it is unknown which exact portion of the total processing time of the jobs in σ will arrive as small jobs. In order to cope with these uncertainties ALG^ε has to construct robust schedules. To this end the number of machines is partitioned into two sets μ_c and μ_r. For the machines of μ_c, the algorithm initially determines a good assignment or *configuration* assuming that at least v_i jobs of job class i will arrive. The machines of μ_r are reserve machines and will be assigned additional large jobs as they arise in σ. Small jobs will always be placed on machines in μ_c. The initial configuration determined for these machines has the property that, no matter how many small jobs arrive, a machine load never exceeds $4/3 + \varepsilon$

We proceed to describe ALG^ε in detail. Let $0 < \varepsilon \leq 1$. Moreover, set $\varepsilon' = \varepsilon/8$. Again we assume w. l. o. g. that, for an incoming job sequence, there holds $\text{OPT}(\mu, \sigma) = 1$. Hence the processing time of any job is upper bounded by 1.

Job classes. A job J_t, $1 \leq t \leq n$, is *small* if $p_t \leq 1/3 + 2\varepsilon'$; otherwise J_t is *large*. We divide the range of possible job processing times into job classes. Let $I_s = (0, 1/3 + 2\varepsilon']$ be the interval containing the processing times of small jobs. Let $\lambda = \lceil \log(\frac{3}{8} + \frac{1}{48\varepsilon'}) \rceil$ and $k = \lambda + 2$, where the logarithm is taken to base 2. For $i = 1, \ldots, k$, let

$$a_i = \max\{\tfrac{1}{3} - 2\varepsilon' + (\tfrac{1}{12} + \tfrac{3}{2}\varepsilon')\tfrac{1}{2^{\lambda+1-i}}, \tfrac{1}{3} + 2\varepsilon'\} \quad \text{and} \quad b_i = \tfrac{1}{3} - 2\varepsilon' + (\tfrac{1}{12} + \tfrac{3}{2}\varepsilon')\tfrac{1}{2^{\lambda-i}}.$$

It is easy to verify that $a_1 = 1/3 + 2\varepsilon'$ and $a_i < b_i$, for $i = 1, \ldots, k$. Furthermore $b_{k-1} = 1/2 + \varepsilon'$ and $b_k = 2/3 + 4\varepsilon'$. For $i = 1, \ldots, k$ define $I_i = (a_i, b_i]$. There holds $\bigcup_{1 \leq i \leq k} I_i = (1/3 + 2\varepsilon', 2/3 + 4\varepsilon']$. Moreover, for $i = 1, \ldots, k-1$, let $I_{k+i} = (2a_i, 2b_i]$. Intuitively, I_{k+i} contains the processing times that are twice as large as those in I_i, $1 \leq i \leq k-1$. There holds $\bigcup_{1 \leq i \leq k-1} I_{k+i} = (2/3 + 4\varepsilon', 1 + 2\varepsilon']$. Hence $I_s \cup I_1 \cup \ldots \cup I_{2k-1} = (0, 1 + 2\varepsilon']$. In the following I_i represents *job class* i, for $i = 1, \ldots, 2k-1$. We say that J_t is a *class-i job* if $p_t \in I_i$, where $1 \leq i \leq 2k - 1$.

Definition of target configurations. As mentioned above, for any incoming job sequence σ, ALG^ε works with estimates on the number of class-i jobs arising in

σ, $1 \leq i \leq 2k-1$. For each estimate, the algorithm initially determines a *target configuration* on a subset of the machines, assuming that the estimated set of large jobs will indeed arrive. Intuitively a target configuration describes how many jobs of which job classes should be assigned to a certain subset of the machines.

Hence we partition the m machines into two sets μ_c and μ_r. Let $\tau = \lceil (1+\varepsilon')/(1+2\varepsilon') \cdot m \rceil$. Moreover, let $\mu_c = \{M_1, \ldots, M_\tau\}$ and $\mu_r = \{M_{\tau+1}, \ldots, M_m\}$. Set μ_c contains the machines for which a target configuration will be computed; μ_r contains the reserve machines. The proportion of $|\mu_r|$ to $|\mu_c|$ is roughly $1 : (1+1/\varepsilon')$.

A target configuration has the important property that any machine $M_j \in \mu_c$ contains large jobs of only one job class i, $1 \leq i \leq 2k-1$. Therefore, a target configuration is properly defined by a vector $c = (c_1, \ldots, c_\tau) \in \{0, \ldots, 2k-1\}^\tau$. If $c_j = 0$, then M_j does not contain any large jobs in the target configuration, $1 \leq j \leq \tau$. If $c_j = i$, where $i \in \{1, \ldots, 2k-1\}$, then M_j contains class-i jobs, $1 \leq j \leq \tau$. The vector c implicitly also specifies how many large jobs reside on a machine. If $c_j = i$ with $1 \leq i \leq k$, then M_j contains two class-i jobs. Note that, for general $i \in \{1, \ldots, k\}$, a third job cannot be placed on the machine without exceeding the load bound of $4/3+\varepsilon$. If $c_j = i$ with $k+1 \leq i \leq 2k-1$, then M_j contains one class-i job. Again, the assignment of a second job is not feasible in general. Given a configuration c, M_j is referred to as a *class-i machine* if $c_j = i$, where $1 \leq j \leq \tau$ and $1 \leq i \leq 2k-1$.

Each vector $c = (c_1, \ldots, c_\tau)$ encodes inputs containing $2|\{c_j \in \{c_1, \ldots c_\tau\} \mid c_j = i\}|$ class-i jobs, for $i = 1, \ldots, k$, as well as $|\{c_j \in \{c_1, \ldots c_\tau\} \mid c_j = i\}|$ class-i jobs, for $i = k+1, \ldots, 2k-1$, which can be seen by the above interpretation of target configurations. Hence, for an incoming job sequence, we can consider target configurations as estimates on the number of class-i jobs, for any $1 \leq i \leq 2k-1$. Unfortunately, it will not be possible to work with all target configurations $c \in \{0, \ldots, 2k-1\}^\tau$ since the resulting number of schedules to be constructed would be $(2k)^\tau = (\log(1/\varepsilon))^{\Omega(m)}$. Therefore, we will work with a suitable sparsification of the set of all configurations.

Sparsifying the set of target configurations. Let $\kappa = \lceil 2(2+1/\varepsilon')(2k-1) \rceil$ and $U = \{0, \ldots, \kappa\}^{2k-1}$. In Section 4.4.2 we will show that $\kappa \lfloor (m-\tau)/(2k-1) \rfloor \geq m$ if m is not too small (see Lemma 4.8). This property in turn will ensure that any job sequence σ can be mapped to a $u \in U$. For any vector $u = (u_1, \ldots, u_{2k-1}) \in U$, we define a target configuration $c(u)$ that contains $u_i \lfloor (m-\tau)/(2k-1) \rfloor$ class-i machines, for $i = 1, \ldots, 2k-1$, provided that $\sum_{i=1}^{2k-1} u_i \lfloor (m-\tau)/(2k-1) \rfloor$ does not exceed τ. For any $u = (u_1, \ldots, u_{2k-1}) \in U$, let $\pi_i = \sum_{j=1}^{i} u_j \lfloor (m-\tau)/(2k-1) \rfloor$, be the partial sums of the first i entries of u, multiplied by $\lfloor (m-\tau)/(2k-1) \rfloor$, for $i = 0, \ldots, 2k-1$. Let $\tau' = \pi_{2k-1}$. First construct a vector $c'(u) = (c'_1, \ldots, c'_{\tau'})$

of length τ' that contains exactly $u_i\lfloor(m-\tau)/(2k-1)\rfloor$ class-i machines. That is, for $i = 1, \ldots, 2k-1$, let $c'_j = i$ for $j = \pi_{i-1}+1, \ldots, \pi_i$. We now truncate or extend $c'(u)$ to obtain a vector of length τ. If $\tau' \geq \tau$, then $c(u)$ is the vector consisting of the first τ entries of $c'(u)$. If $\tau' < \tau$, then $c(u) = (c'_1, \ldots, c'_{\tau'}, 0, \ldots, 0)$, i. e. the last $\tau - \tau'$ entries are set to 0. Let $C = \{c(u) \mid u \in U\}$ be the set of all target configurations constructed from vectors $u \in U$.

Description of ALG$^\varepsilon$. In a first step ALG$^\varepsilon$ enumerates the set C of target configurations. Algorithm ALG$^\varepsilon$ sequences each job J_t in every schedule $S^{(c)}$, for any $c \in C$, using algorithm ALG$_c$. Hereby ALG$_c$ works as follows. ALG$_c$ uses the target configuration specified by $c = (c_1, \ldots, c_\tau)$ for the machines of μ_c to make scheduling decisions. Consider a machine $M_j \in \mu_c$ and suppose $c_j > 0$, i. e. M_j is a class-i machine for some $i \geq 1$.

We define $\ell^-(j)$ and $\ell^+(j)$ to be the *targeted minimal* and *maximal loads* caused by large jobs on M_j, according to the target configuration as follows. If $c_j = i$ for some $i \in \{1, \ldots, k\}$, then let $\ell^-(j) = 2a_i$ and $\ell^+(j) = 2b_i$. Recall that in a target configuration a class-i machine contains two class-i jobs if $1 \leq i \leq k$. If $c_j = i$ for some $i \in \{k+1, \ldots, 2k-1\}$ and hence $i = k + i'$ for some $i' \in \{1, \ldots, k-1\}$, then $\ell^-(j) = 2a_{i'}$ and $\ell^+(j) = 2b_{i'}$. If $M_j \in \mu_c$ is a machine with $c_j = 0$, then $\ell^-(j) = \ell^+(j) = 0$.

Assume that M_j is a class-i machine with $i \geq 1$. If $i \in \{1, \ldots, k\}$, then at any time during the scheduling process we call machine M_j *admissible* if it has received less than two class-i jobs so far. Analogously, if $i \in \{k+1, \ldots, 2k-1\}$, then we call machine M_j *admissible* if it has received no class-i job so far. As usually, at any time during the scheduling process, let $\ell(j)$ be the current load of machine M_j and let $\ell_s(j)$ be the load due to small jobs, $1 \leq j \leq m$.

Algorithm ALG$_c$ schedules each incoming job J_t, $1 \leq t \leq n$, in the following way. First assume that J_t is a large job and, in particular, a class-i job, $1 \leq i \leq 2k-1$. The algorithm checks if there is a class-i machine in μ_c that is admissible. If so, J_t is assigned to such a machine. If there is no admissible class-i machine available, then J_t is placed on a machine in μ_r. There jobs are scheduled according to the *Best-Fit* policy, i. e. ALG$_c$ checks if there exists a machine $M_j \in \mu_r$ such that $\ell(j) + p_t \leq 4/3 + \varepsilon$. If this is the case, then J_t is assigned to such a machine with the largest current load $\ell(j)$. If no such machine exists, J_t is assigned to an arbitrary machine in μ_r. Next assume that J_t is small. Then job J_t is assigned to a machine in μ_c, where preference is given to machines that have already received small jobs. Algorithm ALG$_c$ checks if there is an $M_j \in \mu_c$ with $\ell_s(j) > 0$ such that $\ell^+(j) + \ell_s(j) + p_t \leq 4/3 + \varepsilon$. If this is the case, then J_t is assigned to any such machine. Otherwise ALG$_c$ considers the machines of μ_c which have not yet received any small jobs. If there exists an $M_j \in \mu_c$ with $\ell_s(j) = 0$ such that

$\ell^+(j)+p_t \leq 4/3+\varepsilon$, then among these machines J_t is assigned to one having the smallest targeted load $\ell^-(j)$. If again no such machine exists, J_t is assigned to an arbitrary machine in μ_c. A summary of ALG^ε is given in Figure 4.4, and algorithm ALG_c is depicted in Figure 4.5.

1. Let $\varepsilon' = \varepsilon/8$.
2. Let $k = \lceil \log(\frac{3}{8} + \frac{1}{48\varepsilon'}) \rceil + 2$.
3. Let $\kappa = \lceil 2(2+1/\varepsilon')(2k-1) \rceil$.
4. Let $\tau = \lceil (1+\varepsilon')/(1+2\varepsilon') \cdot m \rceil$.
5. Enumerate the target configurations $C = \{c(u) \mid u \in U\}$, where $U = \{0, \ldots, \kappa\}^{2k-1}$.
6. Sequence each incoming J_t into each schedule $S^{(c)}$ using ALG_c in Figure 4.5, where ALG_c is given access to the target configuration $c(u)$.

Figure 4.4: The algorithm ALG^ε.

Theorem 4.5. ALG^ε *is* $(4/3+\varepsilon)$*-competitive, for any* $0 < \varepsilon \leq 1$ *and* $m \geq 2k/\varepsilon'^2$. *The algorithm uses* $1/\varepsilon^{O(\log(1/\varepsilon))}$ *schedules.*

ALG^ε is $(4/3+\varepsilon)$-competitive if, for the chosen ε, the number of machines is at least $2k/\varepsilon'^2$. If the number of machines is smaller, we can simply apply algorithm PTAS^ε with an accuracy of $\varepsilon = 1/3$. Let $\text{ALG}^{\hat{\varepsilon}}$ be the following combined algorithm. If for the chosen $\hat{\varepsilon}$, $m < 2k/\hat{\varepsilon}'^2$, execute $\text{PTAS}^{1/3}$, where $\hat{\varepsilon}' = \hat{\varepsilon}/8$ as above. Otherwise execute $\text{ALG}^{\hat{\varepsilon}}$.

Corollary 4.6. $\text{ALG}^{\hat{\varepsilon}}$ *is* $(4/3+\hat{\varepsilon})$*-competitive, for any* $0 < \hat{\varepsilon} \leq 1$*, and uses* $1/\hat{\varepsilon}^{O(\log(1/\hat{\varepsilon}))}$ *schedules.*

Proof. If $\text{PTAS}^{1/3}$ is executed for a machine number $m < 2k/\hat{\varepsilon}'^2$, then by Theorem 4.4 the number of schedules is $(\log(1/\hat{\varepsilon})/\hat{\varepsilon}^3)^{O(1)}$, which is $1/\hat{\varepsilon}^{O(1)}$. □

4.4.2 Analysis of the Algorithm

In this section we prove Theorem 4.5. First we will show that, for any job sequence σ, ALG^ε generates a schedule whose makespan is at most $4/3+\varepsilon = (4/3+\varepsilon)\text{OPT}(\mu,\sigma)$. In more detail we will prove that, for any σ, there exists a target configuration $c \in C$ that accurately models the large jobs arising in σ. We will refer to such a vector as a *valid target configuration*. Then we will show that

1. Let J_t be the job to be assigned.
2. If J_t is large, then execute the following.
   (a) Compute index i such that J_t is a class-i job.
   (b) If $\exists M_j \in \mu_c$, and M_j is admissible class-i machine, then assign J_t to M_j.
   (c) Otherwise:
      (i) If $\exists\ M_j \in \mu_r$ such that $\ell(j) + p_t \leq 4/3 + \varepsilon$, then assign J_t to such an M_j with the largest $\ell(j)$;
      (ii) otherwise place J_t on an arbitrary $M_j \in \mu_r$.
3. If J_t is small, then execute the following.
   (a) If $\exists\ M_j \in \mu_c$ with $\ell_s(j) > 0$ such that $\ell^+(j) + \ell_s(j) + p_t \leq 4/3 + \varepsilon$, then assign J_t to such an M_j.
   (b) Otherwise if $\exists\ M_j \in \mu_c$ with $\ell_s(j) = 0$ such that $\ell^+(j) + p_t \leq 4/3 + \varepsilon$, then assign J_t to such an M_j with the lowest $\ell^-(j)$.
   (c) Otherwise place J_t on an arbitrary $M_j \in \mu_c$.

Figure 4.5: The algorithm ALG_c.

the corresponding algorithm ALG_c builds a schedule with a makespan of at most $(4/3 + \varepsilon)\text{OPT}(\mu, \sigma)$.

We introduce some notation. Consider any job sequence σ. For any i, $1 \leq i \leq 2k-1$, let $n_i(\sigma)$ be the number of class-i jobs arising in σ, i. e. $n_i(\sigma)$ is the number of jobs J_t with $p_t \in I_i$. Furthermore, for any target configuration $c = (c_1, \ldots, c_\tau) \in C$ and any i with $1 \leq i \leq 2k-1$, let m_i be the number of class-i machines in c, i. e. $m_i = |\{c_j \in \{c_1, \ldots, c_\tau\} \mid c_j = i\}|$. Let $\tau_1 = \sum_{i=1}^{k} m_i$ be the total number of class-i machines with $i \in \{1, \ldots, k\}$. Similarly, $\tau_2 = \sum_{i=k+1}^{2k-1} m_i$ is the total number of class-i machines with $i \in \{k+1, \ldots, 2k-1\}$. Given σ, vector $c \in C$ will be a valid target configuration if, for any $i = 1, \ldots, 2k-1$, σ contains as many class-i jobs as specified in c and, moreover, if all the additional large jobs can be feasibly scheduled on the $m - \tau$ reserve machines. Recall that in a configuration c, any class-i machine with $1 \leq i \leq k$ is supposed to contain two class-i jobs. Formally, $c \in C$ is a *valid target configuration* if the following three conditions hold.

(i) For $i = 1, \ldots, k$, there holds $2m_i \leq n_i(\sigma)$.

(ii) For $i = k+1, \ldots, 2k-1$, there holds $m_i \leq n_i(\sigma)$.

(iii) There holds $\lceil (\sum_{i=1}^{k} n_i(\sigma) - 2\tau_1)/2 \rceil + \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2 \leq m - \tau$.

Conditions (i) and (ii) represent the constraint that σ contains as many class-i jobs as specified in c, $1 \leq i \leq 2k-1$. Condition (iii) models the requirement that additional large jobs can be feasibly packed on the reserve machines. Here $\sum_{i=1}^{k} n_i(\sigma) - 2\tau_1$ is the extra number of class-i jobs with $i \in \{1, \ldots, k\}$ in σ. Any two of these can be packed on one machine since the processing time of any of these jobs is upper bounded by $b_k \leq 2/3 + 4\varepsilon'$. Hence two jobs incur a machine load of at most $4/3 + 8\varepsilon' = 4/3 + \varepsilon$. Analogously, $\sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2$ is the extra number of class-i jobs with $i \in \{k+1, \ldots, 2k-1\}$, which cannot be combined with other jobs because their processing times are greater than $2a_1 \geq 2/3 + 4\varepsilon'$.

In order to prove that, for any σ, there exists a valid target configuration we need two lemmas.

Lemma 4.7. *For any σ, there holds $\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \rceil + \sum_{i=k+1}^{2k-1} n_i(\sigma) \leq m$.*

Proof. Consider any optimal schedule O for σ and recall that we assume w. l. o. g. that $\text{OPT}(\mu, \sigma) = 1$. In O any machine containing a class-i job with $i \in \{k+1, \ldots, 2k-1\}$ cannot contain an additional large job: The class-i job causes a load greater than $2a_1 \geq 2/3 + 4\varepsilon'$ and any additional large job, having a processing time greater than $1/3 + 2\varepsilon'$, would generate a total load greater than 1. Furthermore, any machine containing a class-i job with $i \in \{1, \ldots, k\}$ can contain at most one additional job of the job classes $1, \ldots, k$ because two further jobs would generate a total load greater than $3a_1 \geq 3(1/3 + 2\varepsilon') > 1$. □

Lemma 4.8. *For any $0 < \varepsilon' \leq 1/8$, there holds $\kappa \lfloor (m-\tau)/(2k-1) \rfloor \geq m$ if $m \geq 2k/{\varepsilon'}^2$.*

Proof. There holds

$$\begin{aligned}
\kappa \lfloor (m-\tau)/(2k-1) \rfloor &\geq 2(2 + \tfrac{1}{\varepsilon'})(2k-1) \cdot \lfloor (m-\tau)/(2k-1) \rfloor \\
&\geq 2(2 + \tfrac{1}{\varepsilon'})(2k-1) \cdot ((m-\tau)/(2k-1) - 1) \\
&= 2(2 + \tfrac{1}{\varepsilon'})(2k-1) \cdot \left(\frac{m - \lceil \frac{1+\varepsilon'}{1+2\varepsilon'} m \rceil}{2k-1} - 1 \right) \\
&\geq 2(2 + \tfrac{1}{\varepsilon'})(2k-1) \cdot \left(\frac{\frac{\varepsilon'}{1+2\varepsilon'} m - 1}{2k-1} - 1 \right) \\
&= 2(2 + \tfrac{1}{\varepsilon'})(2k-1) \cdot \left(\frac{\varepsilon' m - (1+2\varepsilon')2k}{(1+2\varepsilon')(2k-1)} \right)
\end{aligned}$$

$$\geq m + m - (2/\varepsilon')(1 + 2\varepsilon')2k \geq m,$$

where the last inequality follows because of $m \geq 2k/{\varepsilon'}^2$ and $2k/{\varepsilon'}^2 \geq (2/\varepsilon')(1+2\varepsilon')2k$, for any $\varepsilon' \leq 1/8$. □

The next lemma establishes the existence of valid target configurations.

Lemma 4.9. *For any σ, there exists a valid target configuration $c \in C$ if $m \geq 2k/{\varepsilon'}^2$.*

Proof. In this proof let $m_0 = \lfloor (m-\tau)/(2k-1) \rfloor$. Given σ, we first construct a vector $u \in U$. Lemma 4.7 implies that for any job class i, $1 \leq i \leq k$, there holds $\lceil n_i(\sigma)/2 \rceil \leq m$. For any job class i, $k+1 \leq i \leq 2k-1$, there holds $n_i(\sigma) \leq m$. By Lemma 4.8, $\kappa m_0 \geq m$, which is equivalent to $m/m_0 \leq \kappa$. For any i with $1 \leq i \leq k$, set $u_i = \lfloor n_i(\sigma)/(2m_0) \rfloor$. For any i with $k+1 \leq i \leq 2k-1$, set $u_i = \lfloor n_i(\sigma)/m_0 \rfloor$. Then $u_i \in \{0, \ldots, \kappa\}$, for $i = 1, \ldots, 2k-1$, and the resulting vector $u = (u_1, \ldots . u_{2k-1})$ is element of U. We next show that the vector $c(u)$ constructed by ALG$^\varepsilon$ is a valid target configuration.

When ALG$^\varepsilon$ constructs $c(u)$, it first builds a vector $c'(u) = (c'_1, \ldots, c'_{\tau'})$ of length $\tau' = \sum_{i=1}^{2k-1} u_i m_0$ containing exactly $u_i m_0$ entries with $c'_j = i$, for $i = 1, \ldots, 2k-1$. If $\tau' \geq \tau$, then $c(u)$ contains the first τ entries of $c'(u)$. If $\tau' < \tau$, then $c(u)$ is obtained from $c'(u)$ by adding $\tau - \tau'$ entries of value 0. In either case $c(u)$ contains at most $u_i m_0$ entries of values i, for $i = 1, \ldots, 2k-1$. Hence for the target configuration $c(u)$, there holds $m_i \leq u_i m_0$, for $i = 1, \ldots, 2k-1$, where m_i is again the total number of class-i machines in $c(u)$.

If $i \in \{1, \ldots, k\}$, then $m_i \leq \lfloor n_i(\sigma)/(2m_0) \rfloor m_0 \leq n_i(\sigma)/2$, which is equivalent to $2m_i \leq n_i(\sigma)$. Similarly we find $m_i \leq \lfloor n_i(\sigma)/m_0 \rfloor m_0 \leq n_i(\sigma)$ if $i \in \{k+1, \ldots, 2k-1\}$. Therefore, conditions (i) and (ii) defining valid target configurations are satisfied and we are left to verify condition (iii).

First assume $\tau' \geq \tau$. In this case the vector $c(u)$ contains no entries of value 0 and hence $\tau = \tau_1 + \tau_2$. Recall that $\tau_1 = \sum_{i=1}^{k} m_i$ is the total number of class-i machines with $i \in \{1, \ldots, k\}$ specified in $c(u)$. Similarly, $\tau_2 = \sum_{i=k+1}^{2k-1} m_i$ is the total number of class-i machines with $i \in \{k+1, \ldots, 2k-1\}$. By Lemma 4.7, $\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \rceil + \sum_{i=k+1}^{2k-1} n_i(\sigma) \leq m$. Subtracting the equation $\tau_1 + \tau_2 = \tau$, we obtain

$$\left\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \right\rceil - \tau_1 + \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2 \leq m - \tau.$$

There holds $\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \rceil - \tau_1 = \lceil (\sum_{i=1}^{k} n_i(\sigma) - 2\tau_1)/2 \rceil$ because τ_1 is an integer. Hence condition (iii) defining valid target configurations is satisfied.

It remains to consider the case $\tau' < \tau$. For any i with $i \in \{k+1, \ldots, 2k-1\}$, there holds $u_i = \lfloor n_i(\sigma)/m_0 \rfloor$ and hence $u_i > n_i(\sigma)/m_0 - 1$, which is equivalent

to $n_i(\sigma) < (u_i+1)m_0$. Hence

$$\sum_{i=k+1}^{2k-1} n_i(\sigma) < \sum_{i=k+1}^{2k-1} (u_i+1)m_0 = \sum_{i=k+1}^{2k-1} u_i m_0 + (k-1)m_0.$$

The sum $\sum_{i=k+1}^{2k-1} u_i m_0 = \sum_{i=k+1}^{2k-1} u_i \lfloor (m-\tau)/(2k-1) \rfloor$ is the total number of entries c'_j with $c'_j \in \{k+1,\ldots,2k-1\}$ in $c'(u)$. Since $\tau' < \tau$, none of these entries is deleted when $c(u)$ is derived from $c'(u)$. Hence $\sum_{i=k+1}^{2k-1} u_i m_0 = \tau_2$ is the total number of class-i machines with $i \in \{k+1,\ldots,2k-1\}$ specified in $c(u)$. We conclude

$$\sum_{i=k+1}^{2k-1} n_i(\sigma) \leq \tau_2 + (k-1)m_0. \tag{4.6}$$

For any i with $i \in \{1,\ldots,k\}$, there holds $u_i = \lfloor n_i(\sigma)/(2m_0) \rfloor$ and hence $u_i > n_i(\sigma)/(2m_0) - 1$. This implies $n_i(\sigma)/2 < (u_i+1)m_0$. Since $(u_i+1)m_0$ is an integer, we obtain $n_i(\sigma)/2 \leq (u_i+1)m_0 - 1$. Thus

$$\left\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \right\rceil \leq \sum_{i=1}^{k} n_i(\sigma)/2 + 1 \leq \sum_{i=1}^{k} (u_i+1)m_0 = \tau_1 + km_0. \tag{4.7}$$

Again $\sum_{i=1}^{k} u_i m_0 = \tau_1$ because $c'(u)$ contains exactly $\sum_{i=1}^{k} u_i m_0$ entries c'_j with $c'_j \in \{1,\ldots,k\}$ and all of these entries are contained in $c(u)$ representing class-i machines for $i \in \{1,\ldots,k\}$. Inequalities (4.6) and (4.7) together with the identity $m_0 = \lfloor (m-\tau)/(2k-1) \rfloor$ imply

$$\left\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \right\rceil - \tau_1 + \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2 \leq (2k-1)\lfloor (m-\tau)/(2k-1) \rfloor \leq m - \tau.$$

Since again $\lceil \sum_{i=1}^{k} n_i(\sigma)/2 \rceil - \tau_1 = \lceil (\sum_{i=1}^{k} n_i(\sigma) - 2\tau_1)/2 \rceil$, condition (iii) defining valid target configurations holds. □

We next analyze the scheduling steps of ALG^{ε}.

Lemma 4.10. *Let ALG_c be any algorithm of ALG^{ε} processing a job sequence σ. At any time there exists at most one machine $M_j \in \mu_c$ with $\ell_s(j) > 0$ and $\ell^-(j) + \ell_s(j) < 1 + \varepsilon'$ in the schedule maintained by ALG_c.*

Proof. Consider any point in time while ALG_c sequences σ. Suppose that there exists a machine $M_j \in \mu_c$ with $\ell_s(j) > 0$ and $\ell^-(j) + \ell_s(j) < 1 + \varepsilon'$. We show that if a small job J_t arrives and ALG_c assigns it to a machine $M_{j'} \in \mu_c$ with $\ell_s(j) = 0$, then $\ell^-(j') + p_t > 1 + \varepsilon'$ so that no new machine with the property

specified in the lemma is generated. A first observation is that M_j is not a class-k machine because in this case $\ell^-(j)$ would be $2a_k = 2b_{k-1} = 1 + 2\varepsilon'$. Also, if $M_{j'}$ is a class-k machine, there is nothing to show because, again, in this case $\ell^-(j') \geq 1 + 2\varepsilon'$.

So assume that ALG_c assigns J_t to a machine $M_{j'} \in \mu_c$, which is not a class-k machine, and $\ell_s(j') = 0$ prior to the assignment. We first show that $\ell^-(j') \geq \ell^-(j)$. Consider the scheduling step in which ALG_c assigned the first small job $J_{t'}$ to M_j. Since M_j is not a class-k machine, $\ell^+(j) = 2b_i$, for some $i \in \{1, \ldots, k-1\}$, and the assignment of $J_{t'}$ to M_j led to a load of at most $\ell^+(j) + p_{t'} \leq 1 + 2\varepsilon' + 1/3 + 2\varepsilon' = 4/3 + 4\varepsilon' < 4/3 + \varepsilon$. As $M_{j'}$ is not a class-k machine either, $J_{t'}$ could have also been assigned to $M_{j'}$ incurring a resulting load of at most $\ell^+(j') + p_{t'} < 4/3 + \varepsilon$ on this machine. Note that when an algorithm ALG_c cannot assign a small job to a machine $M_j \in \mu_c$ with $\ell_s(j) > 0$ and instead has to resort to machines $M_l \in \mu_c$ with $\ell_s(l) = 0$, it chooses a machine having the smallest $\ell^-(l)$ value. We conclude $\ell^-(j) \leq \ell^-(j')$.

Next consider the assignment of J_t. Algorithm ALG_c would prefer to place J_t on M_j as it already contains small jobs. Since this is impossible, there holds $\ell^+(j) + \ell_s(j) + p_t > 4/3 + \varepsilon$ and thus $p_t > 4/3 + 8\varepsilon' - \ell^+(j) - \ell_s(j)$. Since by assumption $\ell^-(j) + \ell_s(j) < 1 + \varepsilon'$, it follows $p_t > 1/3 + 7\varepsilon' - \ell^+(j) + \ell^-(j)$. Suppose that $\ell^+(j) = 2b_i$, for some $i \in \{1, \ldots, k-1\}$. Then $\ell^-(j) = 2a_i$. Since $\ell^-(j') \geq \ell^-(j)$, we obtain

$$\begin{aligned}
\ell^-(j') + p_t &\geq 1/3 + 7\varepsilon' + \ell^-(j) - \ell^+(j) + \ell^-(j) \\
&\geq 1/3 + 7\varepsilon' + 2\left(\frac{1}{12} + \frac{3}{2}\varepsilon'\right)\left(\frac{1}{2^{\lambda+1-i}} - \frac{1}{2^{\lambda-i}}\right) \\
&\quad + 2/3 - 4\varepsilon' + 2\left(\frac{1}{12} + \frac{3}{2}\varepsilon'\right)\frac{1}{2^{\lambda+1-i}} \\
&= 1 + 3\varepsilon' > 1 + \varepsilon',
\end{aligned}$$

as desired. □

The following lemmas focus on algorithms ALG_c such that c is a valid target configuration for σ.

Lemma 4.11. *Let σ be any job sequence and ALG_c be an algorithm such that c is a valid target configuration for σ. Let $m \geq 2k/\varepsilon'^2$. Consider any point in time during the scheduling process. If the schedule of ALG_c contains at most one machine $M_j \in \mu_c$ with $\ell^-(j) + \ell_s(j) < 1 + \varepsilon'$, then no further small job can arrive.*

Proof. Since c is a valid target configuration for σ, by conditions (i) and (ii) defining valid target configurations, the job sequence contains as many class-i jobs, for

any $i \in \{1,\dots,k\}$, as indicated by c. Hence the total processing time of large jobs in σ is lower bounded by $\sum_{j=1}^{\tau} \ell^-(j)$. Hence the total processing time of jobs in σ is at least $\sum_{j=1}^{\tau}(\ell^-(j)+\ell_s(j))$, where the machine loads due to small jobs may be considered at an arbitrary point in time. Hence if there exists a time such that $\ell_s(j)+\ell^-(j) < 1+\varepsilon'$ for at most one $M_j \in \mu_c$, we obtain

$$\begin{aligned}\sum_{j=1}^{\tau}(\ell^-(j)+\ell_s(j)) &\geq (1+\varepsilon')(\tau-1)\\ &\geq (1+\varepsilon')(\frac{1+\varepsilon'}{1+2\varepsilon'}m-1)\\ &= m+\frac{{\varepsilon'}^2}{1+2\varepsilon'}m-(1+\varepsilon') \geq m.\end{aligned}$$

The last inequality holds because $m \geq 2k/{\varepsilon'}^2 \geq 2/{\varepsilon'}^2 \geq (1+\varepsilon')(2\varepsilon'+1)/{\varepsilon'}^2$, for any $\varepsilon' \leq 1/8$. Hence no further small job can arrive. □

Lemma 4.12. *Let σ be any job sequence and ALG_c be an algorithm such that c is a valid target configuration for σ. Let $m \geq 2k/{\varepsilon'}^2$. Then in the final schedule constructed by ALG_c, each machine in μ_c has a load of at most $4/3+\varepsilon$.*

Proof. We consider the scheduling steps in which ALG_c assigns a job J_t to a machine in μ_c. First suppose that J_t is large. Let J_t be a class-i job, where $1 \leq i \leq 2k-1$. If J_t is assigned to an $M_j \in \mu_c$, then M_j must be an admissible class-i machine, i. e. prior to the assignment of J_t it contains fewer class-i jobs as specified by the target configuration. This implies that for any machine $M_j \in \mu_c$, its load due to large jobs is always at most $\ell^+(j)$. The latter value is upper bounded by $2b_k \leq 2(2/3+4\varepsilon') = 4/3+8\varepsilon' = 4/3+\varepsilon$. Hence, in order to establish the lemma it suffices to show that whenever a small job is assigned to a machine $M_j \in \mu_c$, the resulting load $\ell^+(j)+\ell_s(j)$ on M_j is at most $4/3+\varepsilon$.

Suppose on the contrary that a small job J_t arrives and ALG_c schedules it on a machine in μ_c such that the resulting load is greater than $4/3+\varepsilon$. Algorithm ALG_c first tries to place J_t on a machine $M_j \in \mu_c$ with $\ell_s(j) > 0$, which has already received small jobs. By Lemma 4.10, among these machines there exists at most one having the property that $\ell^-(j)+\ell_s(j) < 1+\varepsilon'$. Since an assignment to those machines is impossible without exceeding a load of $4/3+\varepsilon$, ALG_c tries to place J_t on a machine $M_j \in \mu_c$ with $\ell_s(j) = 0$. Since this is also impossible without exceeding a load of $4/3+\varepsilon$, any $M_j \in \mu_c$ with $\ell_s(j) = 0$ must be a class-k machine. This holds true because for any class-i machine with $i \neq k$, there holds $\ell^+(j) \leq 2b_{k-1} \leq 1+2\varepsilon'$ and an assignment of a small job would result in a total load of at most $1+2\varepsilon'+1/3+2\varepsilon' < 4/3+\varepsilon$. Observe that any class-l machine has a targeted minimal load of $2a_k = 2b_{k-1} \geq 1+2\varepsilon' > 1+\varepsilon'$.

We conclude that immediately before the assignment of J_t the schedule of ALG_c contains at most one machine $M_j \in \mu_c$ with $\ell^-(j) + \ell_s(j) < 1 + \varepsilon'$. Lemma 4.11 implies that the incoming job J_t cannot be small, and we obtain a contradiction. □

Lemma 4.13. *Let σ be any job sequence and ALG_c be an algorithm such that c is a valid target configuration for σ. Then in the final schedule constructed by ALG_c, each machine in μ_r has a load of at most $4/3 + \varepsilon$.*

Proof. Algorithm ALG_c assigns only large jobs to machines in μ_r. A first observation is that whenever there exists an $M_j \in \mu_r$ that contains only one class-i job with $i \in \{1, \ldots, k\}$ but no further jobs, then an incoming class-i' job with $i' \in \{1, \ldots, k\}$ will not be assigned to an empty machine. This holds true because the two jobs can be combined, which results in a total load of at most $2b_k \leq 4/3 + 8\varepsilon' = 4/3 + \varepsilon$.

The observation implies that at any time while ALG_c processes σ, the number of machines of μ_r containing at least one job is upper bounded by $\lceil n_1/2 \rceil + n_2$. Here n_1 denotes the total number of class-i jobs with $i \in \{1, \ldots, k\}$ that have been assigned to machines of μ_r so far. Analogously, n_2 is the total number of class-i jobs with $i \in \{k+1, \ldots, 2k-1\}$ currently residing on machines in μ_r. Since c is a valid target configuration for σ, conditions (i) and (ii) defining these imply $0 \leq \sum_{i=1}^{k} n_i(\sigma) - 2\tau_1$ and $0 \leq \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2$. Thus, since ALG_c assigns large jobs preferably to machines in μ_c, there holds $n_1 \leq \sum_{i=1}^{k} n_i(\sigma) - 2\tau_1$ and $n_2 \leq \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2$. By condition (iii) defining valid target configurations, $\lceil (\sum_{i=1}^{k} n_i(\sigma) - 2\tau_1)/2 \rceil + \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2 \leq m - \tau$. Hence, while $n_2 < \sum_{i=k+1}^{2k-1} n_i(\sigma) - \tau_2$ there holds $\lceil n_1/2 \rceil + n_2 < m - \tau$ and thus exists an empty machine μ_r to which an incoming class-i jobs with $i \in \{k+1, \ldots, 2k-1\}$ can be assigned. Similarly, while $n_1 < \sum_{i=1}^{k} n_i(\sigma) - 2\tau_1$, there must exist an empty machine or a machine containing only one class-i' job with $i' \in \{1, \ldots, k\}$ to which an incoming class-i job with $i \in \{1, \ldots, k\}$ can be assigned. In either case, the assignment generates a load of at most $4/3 + \varepsilon$ on the selected machine. □

Proof. (of Theorem 4.5). Lemma 4.9 ensures that there exists a target configuration $c \in C$ that is valid. Hence, for this specific c we have by Lemma 4.12 that algorithm ALG_c generates a schedule such that for all machines $M_j \in \mu_c$ there holds $\ell(j) \leq 4/3 + \varepsilon$. Lemma 4.13 gives that the same bound on the load holds for the machines $M_j \in \mu_r$.

The stated number of schedules follows from the fact that ALG^{ε} consists of $|C| = (\kappa + 1)^{2k-1}$ algorithms. Recall that $\kappa = \lceil 2(2 + 1/\varepsilon')(2k-1) \rceil$ and $k = \lambda + 2 = \lceil \log(\frac{3}{8} + \frac{1}{48\varepsilon'}) \rceil + 2$. Hence $k = O(\log(1/\varepsilon))$ and $\kappa = O(1/\varepsilon \log(1/\varepsilon))$, which gives that $|C|$ is $1/\varepsilon^{O(\log(1/\varepsilon))}$. □

4.5 Algorithms for MPS

We derive our algorithms for MPS. The strategies are obtained by simply combining RED$^{\varepsilon}$, presented in Section 4.2, with PTAS$^{\varepsilon}$ or ALG$^{\hat{\varepsilon}}$. In order to achieve a precision of ε in the competitive ratio, the strategies are combined with a precision of $\varepsilon/2$ in its parameters. We first derive the algorithm that uses a constant number of schedules.

Let ALG$^{\bar{\varepsilon}}$ and PTAS$^{\bar{\varepsilon}}$ be the algorithms obtained by executing ALG$^{\hat{\varepsilon}/2}$ and PTAS$^{\varepsilon/2}$ in RED$^{\varepsilon/2}$, respectively.

Corollary 4.14. *Let* $0 < \bar{\varepsilon} \leq 1$. ALG$^{\bar{\varepsilon}}$ *is a* $(4/3 + \bar{\varepsilon})$*-competitive algorithm for* MPS *and uses* $1/\bar{\varepsilon}^{O(\log(1/\bar{\varepsilon}))}$ *schedules, for any* $0 < \bar{\varepsilon} \leq 1$.

Proof. Theorem 4.2 and Corollary 4.6 imply that ALG$^{\bar{\varepsilon}}$ is $(4/3 + \bar{\varepsilon})$-competitive, for any $0 < \bar{\varepsilon} \leq 1$, and that the total number of schedules is the product of $1/\bar{\varepsilon}^{O(\log(1/\bar{\varepsilon}))}$ and $\lceil \log(1 + 12\rho/\bar{\varepsilon}) / \log(1 + \bar{\varepsilon}/(6\rho)) \rceil$, where $\rho = 4/3 + \bar{\varepsilon}/2$. By the Taylor series for $\ln(1 + x)$, $-1 < x \leq 1$, we obtain $\ln(1 + x) \geq x/2$, for any $0 < x \leq 1$. Hence the second term of the product is $1/\bar{\varepsilon}^{O(1)}$. □

We next consider the algorithm PTAS$^{\bar{\varepsilon}}$, which achieves a competitive ratio of $1 + \bar{\varepsilon}$.

Corollary 4.15. PTAS$^{\bar{\varepsilon}}$ *is a* $(1 + \bar{\varepsilon})$*-competitive algorithm for* MPS *and uses no more than* $(m/\bar{\varepsilon})^{O(\log(1/\bar{\varepsilon})/\bar{\varepsilon})}$ *schedules, for any* $0 < \bar{\varepsilon} \leq 1$.

Proof. By Theorems 4.2 and 4.4 algorithm PTAS$^{\bar{\varepsilon}}$ is $(1 + \bar{\varepsilon})$-competitive, for any $0 < \bar{\varepsilon} \leq 1$. The total number of schedules is the product of $(\lfloor 4m/\bar{\varepsilon} \rfloor + 1)^{\lceil \log(4/\bar{\varepsilon}) / \log(1+\bar{\varepsilon}/4) \rceil}$ and $\lceil \log(1 + 12\bar{\rho}/\varepsilon) / \log(1 + \varepsilon/(6\rho)) \rceil$, where $\rho = 1 + \bar{\varepsilon}/2$. Again, by the Taylor series, $\ln(1 + x) \geq x/2$, for any $0 < x \leq 1$. Hence both terms of the product are upper bounded by $(m/\bar{\varepsilon})^{O(\log(1/\bar{\varepsilon})/\bar{\varepsilon})}$. □

4.6 Matching Lower Bounds

We give two lower bounds for MPSO, which clearly transfer to MPS.

Theorem 4.16. *Let* ALG *be a deterministic online algorithm for* MPSO. *If* ALG *achieves a competitive ratio smaller than* $4/3$*, then it must maintain at least* $\lfloor m/3 \rfloor + 1$ *schedules.*

Proof. Let ALG be any deterministic online algorithm for MPSO that maintains at most $\lfloor m/3 \rfloor$ schedules. We show that ALG's competitive ratio is at least $4/3$. To this end we construct an adversarial job sequence σ, with OPT$(\mu, \sigma) = 1$, such that

each schedule maintained by ALG has a makespan of at least $4/3 \cdot \text{OPT}(\mu, \sigma) = 4/3$.

The job sequence σ is composed of two subsequences σ_1 and σ_2, i. e. $\sigma = \sigma_1\sigma_2$. Subsequence σ_1 consists of m jobs of processing time $1/3$ each. Subsequence σ_2 will consist of jobs having a processing time of either 2/3 or 1. The exact number of these jobs depends on the schedules constructed by ALG and will be determined later.

Consider the schedules that ALG may have built after all jobs of σ_1 have been assigned. Each such schedule contains m jobs of processing time $1/3$. Suppose that in some schedule S each machine contains either zero, one or three jobs, i. e. there exists no machine in S containing two or more than three jobs. Such a schedule S can be represented by a pair (m_1, m_3), where m_1 denotes the number of machines containing exactly one job and m_3 is the number of machines containing three jobs. Here m_1 and m_3 are non-negative integers such that $m_1 + 3m_3 = m$. Let $P = \{(m_1, m_3) \mid m_1, m_3 \in \mathbb{N}_0 \text{ and } m_1 + 3m_3 = m\}$ be the set of all these pairs. Set P has $\lfloor m/3 \rfloor + 1$ elements because m_3 can take any value between 0 and $\lfloor m/3 \rfloor$ and $m_1 = m - 3m_3$. Let S be an arbitrary schedule containing m jobs of processing time 1/3 and $(m_1, m_3) \in P$. We say that S is an (m_1, m_3)-*schedule* if the number of machines containing exactly one job equals m_1 and the number of machines containing exactly three jobs equals m_3.

Let $\mathcal{S}$ be the set of schedules constructed by ALG when the entire subsequence σ_1 has arrived. By assumption ALG maintains at most $\lfloor m/3 \rfloor$ schedules, i.e. $|\mathcal{S}| \leq \lfloor m/3 \rfloor$. Hence there must exist a pair $(m_1^*, m_3^*) \in P$ such that no schedule of $\mathcal{S}$ is an (m_1^*, m_3^*)-schedule. On the other hand, let S^* be an (m_1^*, m_3^*)-schedule. Assume w. l. o. g. that the machines in S^* are numbered such that $\ell(S_1^*) \leq \ldots \leq \ell(S_m^*)$. Schedule S^* contains $m - m_3^*$ machines with a load smaller than 1 and, in particular, $m - m_1^* - m_3^*$ empty machines.

Now the subsequence σ_2 consists of $m - m_3^*$ jobs, where the j-th job has a processing time of $1 - \ell(S_j^*)$, for $j = 1, \ldots, m - m_3^*$. Hence σ_2 contains $m - m_1^* - m_3^*$ jobs of processing time 1 followed by n_1^* jobs of processing time $2/3$. Clearly, the makespan of an optimal schedule for σ is 1: The jobs of σ_1 are sequenced as in the (m_1^*, m_3^*)-schedule S^*. Obviously, while σ_2 arrives, the j-th job can be assigned to machine M_j in S^*, having a load of $\ell(S_j^*)$, for $j = 1, \ldots, m - m_3^*$, and resulting in a load of 1 after the assignment of the job.

In the remainder of this proof we consider any schedule $S \in \mathcal{S}$ and show that after σ_2 has been sequenced, the resulting makespan is at least 4/3. This establishes the theorem. So let $S \in \mathcal{S}$ be any schedule and recall that S contains m jobs of processing time 1/3 each. If in S there exists a machine that contains at least four of these jobs, then the makespan is already 4/3 and there is nothing to show. Therefore, we restrict ourselves to the case that every machine in S contains at most three jobs. Again we assume w. l. o. g. that the machines in S satisfy

$\ell(S_1) \leq \ldots \leq \ell(S_m)$. Consider the (m_1^*, m_3^*)-schedule S^*. There must exist a machine M_{j^*}, $1 \leq j^* \leq m$, such that $\ell(S_{j^*}) > \ell(S_{j^*}^*)$: If $\ell(S_{j^*}) \leq \ell(S_{j^*}^*)$ held for all $j = 1, \ldots, m$, then $\ell(S_{j^*}) = \ell(S_{j^*}^*)$ for all $j = 1, \ldots, m$ because S and S^* both contain jobs with a total processing time of $m/3$. Thus S would be an (m_1^*, m_3^*)-schedule and we obtain a contradiction. The m_3^* most loaded machines in S^* have a load of 1. It follows that $j^* \leq m - m_3^*$ because otherwise M_{j^*} in S contained at least four jobs. The property $\ell(S_j^*) > \ell(S_{j^*}^*)$ implies $\ell(S_{j^*}) \geq \ell(S_{j^*}^*) + 1/3$ because S and S^* only contain jobs of processing time $1/3$.

We finally show that sequencing of σ_2 leads to a makespan of at least $4/3$ in S. If ALG assigns two jobs of σ_2 to the same machine, then the resulting machine load is at least 4/3 because each job of σ_2 has a processing time of at least $2/3$. So assume that ALG assigns the jobs of σ_2 to distinct machines. The first j^* jobs of σ_2 each have a processing time of at least $1 - \ell(S_{j^*}^*)$ because the jobs arrive in order of non-increasing processing times. In S there exist at most $j^* - 1$ machines having a load strictly smaller than $\ell(S_{j^*})$. Hence, after the first j^* jobs have been scheduled in S, there exists a machine having a load of at least $\ell(S_{j^*}) + 1 - \ell(S_{j^*}^*) \geq \ell(S_{j^*}^*) + 1/3 + 1 - \ell(S_{j^*}^*) = 4/3$. This concludes the proof. □

We next present a lower bound on the number of schedules required by a $(1+\varepsilon)$-competitive algorithm, where $0 < \varepsilon < 1/4$. It implies that, for any fixed ε, the number asymptotically depends on $m^{\Omega(1/\varepsilon)}$, as m increases. For instance, any algorithm with a competitive ratio smaller than $1 + 1/12$ requires $\Omega(m^2)$ schedules. Any algorithm with a competitive ratio smaller than $1 + 1/16$ needs $\Omega(m^3)$ schedules.

Theorem 4.17. *Let* ALG *be a deterministic online algorithm for* MPSO. *If* ALG *achieves a competitive ratio smaller than* $1 + \varepsilon$, *where* $0 < \varepsilon \leq 1/4$, *then it must maintain at least* $\binom{m'+h-1}{h-1}$ *schedules, where* $m' = \lfloor m/2 \rfloor$ *and* $h = \lfloor 1/(4\varepsilon) \rfloor$. *The binomial coefficient increases as* ε *decreases. Its value is at least* $\Omega((\varepsilon m)^{\lfloor 1/(4\varepsilon) \rfloor - 1/2}/\sqrt{m})$.

Proof. We extend the proof of Theorem 4.16. Let $0 < \varepsilon \leq 1/4$. Furthermore, let m' and h be defined as in the theorem statement. There holds $h \geq 1$. Let $\varepsilon' = 1/(4h)$ and note that $\varepsilon' \geq \varepsilon$. We will define a set P whose cardinality is at least $\binom{m'+h-1}{h-1}$, and show that if ALG maintains less than $|P|$ schedules, then its competitive ratio is at least $1 + \varepsilon'$.

We specify a job sequence σ and first assume that m is even. Later we will describe how to adapt σ if m is odd. Again σ is composed of two partial sequences σ_1 and σ_2 so that $\sigma = \sigma_1\sigma_2$. Subsequence σ_1 consists of mh jobs of processing time ε' each. Subsequence σ_2 depends on the schedules constructed by ALG and will be specified below. Consider the possible schedules after σ_1 has been sequenced on

the m machines. We restrict ourselves to schedules having the following property. Each machine has a load of exactly 1 or a load that is at most $1/2-\varepsilon'$. Observe that each machine of load 1 contains $1/\varepsilon'$ jobs. Each machine of load at most $1/2-\varepsilon'$ contains up to $2h-1$ jobs because $(2h-1)\varepsilon' = 2h/(4h) - \varepsilon' = 1/2 - \varepsilon'$. Therefore any schedule with the stated property can be described by a vector $\vec{m} = (m_0, \ldots, m_{2h})$, where m_{2h} is the number of machines having a load of 1 and m_i is the number of machines containing exactly i jobs, for $i = 0, \ldots, 2h-1$. The vector $\vec{m}$ satisfies $\sum_{i=0}^{2h} m_i = m$ and $(1/\varepsilon')m_{2h} + \sum_{i=1}^{2h-1} i m_i = mh$. The last equation specifies the constraint that the schedule contains mh jobs. Let P be the set of all these vectors, i. e. let

$$P = \{(m_0, \ldots, m_{2h}) \mid m_i \in \mathbb{N}_0,\ \sum_{i=0}^{2h} m_i = m,\ (1/\varepsilon')m_{2h} + \sum_{i=1}^{2h-1} i m_i = mh\}.$$

We remark that each $\vec{m} \in P$ uniquely identifies one schedule with our desired property. Let S be any schedule containing exactly mh jobs of processing time ε' and $\vec{m} = (m_0, \ldots, m_{2h}) \in P$. We say that S is an *$\vec{m}$-schedule* if in S there exist m_{2h} machines of load 1 and m_i machines containing exactly i jobs, for $i = 0, \ldots, 2h-1$.

Now suppose that ALG maintains less than $|P|$ schedules. Let $\mathcal{S}$ be the set of schedules constructed by ALG after all jobs of σ_1 have arrived. Since $|\mathcal{S}| < |P|$, there must exist an $\vec{m}^* = (m_0^*, \ldots, m_{2h}^*) \in P$ such that no schedule of $\mathcal{S}$ is an $\vec{m}^*$-schedule. Let S^* be an $\vec{m}^*$-schedule and assume w. l. o. g. that the machines are numbered in S^* such that $\ell(S_1^*) \leq \ldots \leq \ell(S_m^*)$. Subsequence σ_2 consists of $m - m_{2h}^*$ jobs, where the j-th job has a processing time of $1 - \ell(S_j^*)$, for $j = 1, \ldots, m - m_{2h}^*$. Hence σ_2 consists of m_i^* jobs of processing time $1 - i\varepsilon'$, for $i = 0, \ldots, 2h-1$. These jobs arrive in order of non-increasing processing time. Each job has a processing time of at least $1/2 + \varepsilon'$ because $1 - (2h-1)\varepsilon' = 1 - (2h/4h - \varepsilon') = 1/2 + \varepsilon'$. The makespan of an optimal schedule for σ is 1. The jobs of σ_1 are sequenced as in S^*. Then, while the jobs of σ_2 arrive, the j-th job of the subsequence is assigned to machine M_j in S^*, for $1 \leq j \leq m - m_{2h}^*$.

We next show that after ALG has sequenced σ_2, each of its schedules has a makespan of at least $1 + \varepsilon'$. So consider any $S \in \mathcal{S}$ and, as always, we assume w. l. o. g. that $\ell(S_1) \leq \ldots \leq \ell(S_m)$. If in S there exists a machine that has a load of at least $1 + \varepsilon'$ and hence contains at least $1/\varepsilon' + 1$ jobs, then there is nothing to show. So assume that each machine in S contains at most $1/\varepsilon'$ jobs and thus has a load of at most 1. We study the assignment of the jobs of σ_2 to S. If ALG places two jobs of σ_2 on the same machine, then we are done because each job has a processing time of at least $1/2 + \varepsilon'$. Therefore we concentrate on the case that ALG assigns the jobs of σ_2 to distinct machines.

Schedules S and S^* both contain jobs of total processing time $mh\varepsilon'$. Since S is

not an $\vec{m}^*$-schedule, there must exist a j^*, $1 \leq j^* \leq m$, such that $\ell(S_{j^*}) > \ell(S^*_{j^*})$ and hence $\ell(S_{j^*}) \geq \ell(S^*_{j^*}) + \varepsilon'$. Each machine in S has a load of at most 1 while the last $m - m^*_{2h}$ machines in S^* have a load of exactly 1. This implies $j^* \leq m - m^*_{2h}$. The first j^* jobs of σ_2 each have a processing time of at least $1 - \ell(S^*_{j^*})$. However, there exist at most $j^* - 1$ machines in S having a load strictly smaller than $\ell(S^*_{j^*})$. Hence after ALG has sequenced the first j^* jobs of σ_2 there must exist a machine in S with a load of at least $\ell(S_{j^*}) + 1 - \ell(S^*_{j^*}) \geq \ell(S^*_{j^*}) + \varepsilon' + 1 - \ell(S^*_{j^*}) = 1 + \varepsilon'$.

So far we have assumed that m is even. If m is odd, we can easily modify σ. The first job of σ is a job of processing time 1. Then σ_1 and σ_2 follow. These subsequences are defined as above, where m is replaced by the even number $m-1$. In this case

$$P = \{(m_0, \ldots, m_{2h}) \mid m_i \in \mathbb{N}_0,\ \sum_{i=0}^{2h} m_i = m-1,\ m_{2h}/\varepsilon' + \sum_{i=1}^{2h-1} i m_i = (m-1)h\}.$$

The analysis presented above carries over because the first job of σ, having a processing time of 1, must be scheduled on a separate machine and cannot be combined with any job of σ_1 or σ_2 if a competitive ratio smaller than $1 + \varepsilon'$ is to be attained.

We next lower bound the cardinality of P. Again we first focus on the case that m is even. In the definition of P the critical constraint is $(1/\varepsilon')m_{2h} + \sum_{i=1}^{2h-1} i m_i = mh$, which implies that not every vector of $\{0, \ldots, m\}^{2h+1}$ represents a schedule that can be built of mh jobs. In particular, the vector $(0, \ldots, 0, m)$ of length $2h+1$ would require $m/\varepsilon' = 4h$ jobs to be contained in σ_1. Therefore, we introduce a set P' and show $|P'| \leq |P|$. Set P' contains vectors of length $2h + 1$ in which the first $h + 1$ entries as well as the last one are equal to 0. The other entries sum to at most $m/2$, i. e. let

$$P' = \{(0, \ldots, 0, m'_{h+1}, \ldots, m'_{2h-1}, 0) \mid m'_i \in \mathbb{N}_0\ \ \sum_{i=1}^{h-1} m'_{h+i} \leq m/2\}.$$

We show that each $\vec{m}' \in P'$ can be mapped to a $\vec{m} \in P$. The mapping is injective, which implies $|P'| \leq |P|$.

Consider any $\vec{m}' = (0, \ldots, 0, m'_{h+1}, \ldots, m'_{2h-1}, 0) \in P'$. Define the vector $\vec{m} = (m_0, \ldots, m_{2h})$ as follows. For $i = h + 1, \ldots, 2h$, let $m_i = m'_i$. For $i = 0, \ldots, h - 1$, let $m_i = m_{2h-i}$. Finally, let $m_h = m - 2\sum_{i=1}^{h-1} m_i$. Note that $m_0 = m_{2h} = 0$. We next show that $\vec{m} \in P$. There holds

$$\sum_{i=0}^{2h} m_i = \sum_{i=1}^{2h-1} m_i = 2\sum_{i=1}^{h-1} m_i + m_h = m.$$

Furthermore,

$$m_{2h}/\varepsilon' + \sum_{i=0}^{2h-1} i m_i = \sum_{i=1}^{2h-1} i m_i = \sum_{i=1}^{h-1} (i + 2h - i) m_i + h m_h$$
$$= 2h \sum_{i=1}^{h-1} m_i + h(m - 2\sum_{i=1}^{h-1} m_i) = mh.$$

It follows, as desired, $\vec{m} \in P$. Note that the last h entries of $\vec{m}$ are identical to the last h entries of $\vec{m}'$. Hence no two vectors of P' that differ in at least one entry are mapped to the same vector of P. Hence $|P'| \leq |P|$. If the number m of machines is odd, then in the definition of P' the entries of a vector sum to at most $(m-1)/2$. The rest of the construction and analysis is the same. Thus, for a general number m of machines

$$P' = \{(0, \ldots, 0, m'_{h+1}, \ldots, m'_{2h-1}, 0) \mid m'_i \in \mathbb{N}_0, \ \sum_{i=1}^{h-1} m'_{h+i} \leq \lfloor m/2 \rfloor\}.$$

This set contains exactly $\binom{m'+h-1}{h-1}$ elements, where $m' = \lfloor m/2 \rfloor$. We lower bound this binomial coefficient.

For any $k \in \mathbb{N}$, there holds $\sqrt{2\pi e}(k/e)^{k+1/2} \leq k! \leq 2\sqrt{2\pi e}(k/e)^{k+1/2}$, by Stirling's approximation [29]. Hence

$$\begin{aligned}
\binom{m'+h-1}{h-1} &= \frac{(m'+h-1)!}{m'!(h-1)!} \geq \frac{(m'+h-1)^{m'+h-1/2}}{4\sqrt{2\pi}(m')^{m'+1/2}(h-1)^{h-1/2}} \\
&= \frac{1}{4\sqrt{2\pi m'}} \left(1 + \frac{h-1}{m'}\right)^{m'} \left(1 + \frac{m'}{h-1}\right)^{h-1/2} \\
&> \frac{1}{4\sqrt{2\pi m'}} \left(1 + \frac{m/2 - 1/2}{1/(4\varepsilon)}\right)^{h-1/2}.
\end{aligned}$$

The last expression is $\Omega((\varepsilon m)^{\lfloor 1/(4\varepsilon) \rfloor - 1/2}/\sqrt{m})$. □

Chapter 5

Approximation Algorithms for Generalized and Variable-Sized Bin Covering

In this chapter we study generalizations of the NP-hard classical BIN COVERING problem in the offline setting.

5.1 Introduction

The Model Recall that in the unit supply model of GENERALIZED BIN COVERING we are given a set $\mu = \{M_1, \ldots, M_m\}$ of *individual bins* and each bin has a *revenue* r_j and a *demand* d_j. In the infinite supply model there are *arbitrarily many bins* of each type M_j available. Also remember that we denote the set of items by $\sigma = \{J_1, \ldots, J_n\}$ and define that each item $J_t \in \sigma$ has a *size* p_t. Throughout the whole chapter, unless stated otherwise, it will be assumed that m is the number of bins or bin types and n is the number of items in the instance. Recall that a bin is *covered* or *filled* if the total size of the packed items is at least the demand d_j of the respective bin, in which case we earn revenue r_j. The goal is to maximize the total obtained revenue.

Related Work The special case with $r_j = d_j$ is known as VARIABLE-SIZED BIN COVERING. The special case with $r_j = d_j = 1$ is the classical BIN COVERING problem. As already mentioned in Section 1.3.2, to the best of our knowledge, all of the previous work considers the (VARIABLE-SIZED) BIN COVERING problem in the infinite supply model. Surveys on offline and online versions of these problems are given by Csirik and Frenk [21] and by Csirik and Woeginger [25]. Historically, research (on the offline version) of the BIN COVERING problem was

initiated by Assmann et al. [11]. They proved that NEXT FIT is a 2-approximation algorithm. Furthermore, they proved that FIRST FIT DECREASING is an asymptotic $3/2$-approximation and even improved on this result by giving an asymptotic $4/3$-approximate algorithm. Csirik et al. [22] also obtained asymptotic approximation guarantees of $3/2$ and $4/3$ with simpler heuristic algorithms. The next breakthrough was made by Csirik, Johnson, and Kenyon [23] by giving an APTAS for the classical BIN COVERING problem. The algorithm is based on a suitable linear program relaxation and a rounding scheme. Later on, Jansen and Solis-Oba [42] improved upon the running time and gave an AFPTAS. They reduce the number of variables by approximating the linear program formulation of Csirik et al. [23], which yields the desired speed-up. Csirik and Totik [24] gave a lower bound of 2 for every online algorithm for online (VARIABLE-SIZED) BIN COVERING, i. e. when items arrive one by one. Their result holds also asymptotically. Clearly, algorithm NEXT FIT using only the largest bin type is already an asymptotic 2-approximation. Woeginger and Zhang [58] and Epstein [27] give optimal instance depended bounds for online VARIABLE-SIZED BIN COVERING.

Our Contribution In terms of results, we make the following contributions. In Section 5.2 we consider GENERALIZED BIN COVERING in the unit supply model. Our first main result is a 5-approximation algorithm with running time $O(nm\sqrt{m+n})$ in Theorem 5.3. The basic idea is to define an algorithm for a modified version of the problem. Even though this solution may not be feasible for the original problem, it will enable us to provide a good solution for the original problem.

For VARIABLE-SIZED BIN COVERING in the infinite supply model, it is not hard to see that any reasonable algorithm (using only the largest bin type) is an asymptotic 2-approximation. The situation changes considerably in the *unit* supply model: Firstly, limitations in bin availability have to be respected. Secondly, the desired approximation guarantees are non-asymptotic. Our main result here is a tight analysis of the natural and fast NEXT FIT DECREASING (NFD) algorithm for VARIABLE-SIZED BIN COVERING in the unit supply model, which can be found in Section 5.3. Theorem 5.10 states that NFD yields an approximation ratio of at most $9/4 = 2.25$ with running time $O(n \log n + m \log m)$. The approximation guarantee is tight for the algorithm, see Example 5.9. The main idea behind our analysis is to classify bins according to their coverage: The bins that NFD covers with single items are – in some sense – optimally covered. If a bin is covered with at least two items, then their total size is at most twice the demand of the covered bin. Hence those bins yield at least half the achievable revenue. Intuitively, the problematic bins are those that are not covered by NFD: An optimal algorithm might recombine leftover items of NFD with other items to cover

some of these bins and increase the revenue gained. Our analysis gives insight into the limited extend to which such recombinations can be profitable. Firstly, our result is interesting in its own right since NFD is a natural and fast algorithm. Secondly, it is also close to being best possible, in the following sense. A folklore reduction from PARTITION yields that even the classical BIN COVERING problem is not approximable within a factor of two, unless P = NP. This clearly excludes the possibility of a PTAS for BIN COVERING in any of the models. The reduction crucially uses that there are only two identical bins in the BIN COVERING instance it creates. Thus the question arises if one can improve in an asymptotic notion, where the revenue in an optimal solution diverges. Indeed, for the classical BIN COVERING problem with *infinite supply*, there is an A(F)PTAS [23, 42].

Since we have individual bins rather than bin types in the *unit supply* model, there are difficulties for defining a meaningful asymptotics for VARIABLE-SIZED BIN COVERING therein. We discuss this issue in more detail before Theorem 5.28. In Theorem 5.28 we show that, even if there are $m > 2$ bins available and the revenue of an optimal solution diverges, there are instances for which no algorithm can have an approximation ratio smaller than $2 - \varepsilon$, for an asymptotically vanishing $\varepsilon > 0$, unless P = NP. Intuitively, we show that, even in this asymptotic notion, one still has to solve a PARTITION instance on two "large" bins. Hence, for this asymptotics, there is no APTAS for VARIABLE-SIZED BIN COVERING in the *unit* supply model, unless P = NP. However, this fact does *not* exclude the possibility of an A(F)PTAS for VARIABLE-SIZED BIN COVERING in the *infinite* supply model. Indeed, we can give an A(F)PTAS for VARIABLE-SIZED BIN COVERING with infinite supply. Our algorithm is an extension of the APTAS of Csirik et al. [23] for classical BIN COVERING. We remove bin types with small demands and adjust the linear program formulation and the rounding scheme used by [23]. The running-time of the APTAS can then be further improved using the involved method of Jansen and Solis-Oba [42] to yield the claimed AFPTAS in Theorem 5.35.

5.2 Generalized Bin Covering in the Unit Supply Model

In this section we present a 5-approximation for GENERALIZED BIN COVERING in the unit supply model. It is not hard to see that naive greedy strategies that assign items to bins with the largest ratio of revenue to demand or that assign items to bins with largest revenue do not yield a constant approximation ratio. This is shown by the following examples.

Example 5.1. *The following instance* (μ_1, σ_1) *shows that assigning items to bins with largest ratio of revenue to demand does not yield a bounded approximation ratio.* $\mu_1 = \{M_1, M_2\}$, $\sigma_1 = \{J_1\}$ *with* $d_1 = 1$, $r_1 = 1 + \varepsilon$, $d_2 = r_2 = m$ *and* $p_1 = m$. *We have* $\text{OPT}(\mu_1, \sigma_1) = m$ *whereas* $\text{ALG}(\mu_1, \sigma_1) = 1 + \varepsilon$.

Example 5.2. *The following instance* (μ_2, σ_2) *shows that assigning items to bins with the largest revenue does not yield a bounded approximation ratio.* $\mu_2 = \{M_1, \ldots, M_{m+1}\}$, $\sigma_1 = \{J_1, \ldots, J_m\}$ *with* $d_1 = m$, $r_1 = 1 + \varepsilon$, $d_2 = \cdots = d_{m+1} = 1$, $r_2 = \ldots, r_{m+1} = 1$ *and* $p_1 = \cdots = p_m = 1$. *We have* $\text{OPT}(\mu_2, \sigma_2) = m$ *and* $\text{ALG}(\mu_2, \sigma_2) = 1 + \varepsilon$.

Outline of the Algorithm. At the heart of our analysis of the algorithm lies the following observation. In an optimal solution either a not too small fraction of bins is covered by using only one item exceeding the demand of the respective bin or a large fraction of bins is covered by more than one item, and all these items are at most as large as the demand of the bin they have been assigned to. It is easy to see (cf. Observation 5.4) that a bipartite maximum matching gives a solution being at least as good as the partial optimal solution in which the bins are covered with only one item. The more difficult case we have to handle is the case when a large fraction of bins is covered with several items in an optimal solution. We tackle this difficulty by considering an appropriately modified BIN COVERING problem. In this problem, items are only allowed to be assigned to bins with demand of at most their size. We say that the items are admissible to the respective bins. Furthermore, we are allowed to split items into parts and these parts may be distributed among the bins to which the whole item is admissible. Intuitively, in this modified problem the revenue gained for a bin is the fraction of demand covered multiplied with the revenue of the respective bin. In Lemma 5.5 we show that the modified problem can be solved optimally in polynomial time by a greedy algorithm ALG^*.

Via a series of transformations, described in Lemmas 5.7 and 5.8, we construct a solution for the GENERALIZED BIN COVERING problem. We show that either this solution has the desired approximation guarantee or there is one bin with large revenue in the instance such that covering only this single bin already suffices to yield a good approximation guarantee. We obtain the following result.

Theorem 5.3. *There exists a* 5*-approximation for* GENERALIZED BIN COVERING *in the unit supply model, which has running time* $O(nm\sqrt{m+n})$.

Introductory Definitions Recall that any assignment of items to bins is called a solution, and we describe a solution S by a tuple $S = (S_1, \ldots, S_m)$, where $S_j \subseteq \sigma$ is the subset of items assigned in solution S to bin M_j. Notice that the sets S_j have

to be pairwise disjoint. Remember that we define $\ell(S_j) = \sum_{t:J_t \in S_j} p_t$ to be the *load* of bin M_j. When it is clear from context which Solution S is meant, we use as a shorthand $\ell(j) := \ell(S_j)$. For a solution S we define $r(S) = \sum_{j:\ell(j) \geq d_j} r_j$ to be its revenue.

During the analysis we can assume that there are no unassigned items in any solution, which is formally $S_1 \cup \cdots \cup S_m = \sigma$. This is justified since we could add a "useless" bin M_{m+1} with $r_{m+1} = 0$ and $d_{m+1} = \infty$ for sake of analysis.

A covered bin M_j is said to be covered *singularly* if $S_j = \{J_t\}$ for some $1 \leq t \leq n$ with $p_t > d_j$; otherwise it is said to be covered *regularly*. Since we can assume that all items can be assigned to bins, we can also make the following assumptions. A bin M_j that contains an item J_t with $p_t > d_j$ is singularly covered. Notice that, for a bin M_j that is covered regularly, we can assume $\ell(j) \leq 2d_j$. This can be assumed to hold true since bin M_j does not contain an item J_t with $p_t > d_j$, and hence, in case $\ell(j) > 2d_j$, we could remove an item and bin M_j still would be covered.

The following observation settles the case when in an optimal solution a large fraction of bins is covered singularly.

Observation 5.4. *Let (μ, σ) be an instance and fix an optimal solution O to this instance. Let μ_S be the set of bins covered singularly in O and σ_S the set of items assigned to these bins. There is an algorithm* ALG *such that* $\text{ALG}(\mu, \sigma) \geq \text{OPT}(\mu_S, \sigma_S)$. *The running time is $O(nm\sqrt{m+n})$.*

Proof. We define an algorithm ALG that solves the following instance for MAXIMUM WEIGHT BIPARTITE MATCHING optimally: Define a bipartite graph $G = (\mu \cup \sigma, E)$ with edges $E = \{jt \mid M_j \in \mu, J_t \in \sigma : p_t > d_j\}$ and a weight function $w : E \to \mathbb{R}$ given by $w_{jt} = r_j$ for $jt \in E$. Our algorithm ALG determines a MAXIMUM WEIGHT BIPARTITE MATCHING $M \subseteq E$. Since our graph has $m + n$ nodes and at most mn edges, the algorithm of Hopcroft and Karp [39] gives a solution in time $O(nm\sqrt{m+n})$. The induced solution $S = (S_1, \ldots, S_m)$ is $S_j = \{J_t\}$ if $jt \in M$ and $S_j = \emptyset$ otherwise.

Clearly, the singularly covered bins μ_S and the items σ_S assigned to them correspond to a matching in G and vice versa. Thus $\text{ALG}(\mu, \sigma) \geq \text{ALG}(\mu_S, \sigma_S) = \text{OPT}(\mu_S, \sigma_S)$ by the correctness of the matching algorithm. □

Consider the following modified BIN COVERING problem. An item J_t may be split by an algorithm into $n_t \geq 1$ *parts*. We will refer to such an item J_t as n_t many items $J_{t,1}, \ldots J_{t,n_t}$ of positive size. We refer to the $J_{t,l}$ as the *parts of the item* J_t. Let $p_{t,l}$ denote the size of item part $J_{t,l}$ of item J_t. Formally, it has to hold $p_t = \sum_{l=1}^{n_t} p_{t,l}$ and $p_{t,l} > 0$ for $1 \leq l \leq n_t$.

An item J_t is said to be *admissible* to a bin M_j if $p_t \leq d_j$. The parts $J_{t,l}$ of an item J_t are defined to be *admissible* to M_j if and only if J_t is admissible to M_j.

Item parts can only be assigned to bins to which they are admissible.

For a fixed solution $S = (S_1, \ldots, S_m)$ let S_j be the set of item parts assigned to M_j. Let $y_j := \min\{\ell(j)/d_j, 1\}$ be the *fill level* of bin M_j (note that the fill level of bin M_j may be at most one, but nevertheless $\ell(j) > d_j$ is permitted, i. e. the sum of item sizes assigned to bin M_j may exceed the demand of M_j). The revenue gained for bin M_j in the modified problem is $r^*(S_j) := r_j y_j$, which is intuitively the percentage of covered demand multiplied with the revenue of the bin, where the maximal revenue that can be gained is bounded by r_j. Further, for a solution $S = (S_1, \ldots, S_m)$, let $r^*(S) = \sum_{1 \leq j \leq m} r^*(S_j)$. The goal is to find a solution S that maximizes $r^*(S)$ on a given instance (μ, σ). Let the *efficiency* e_j of bin M_j be $e_j := r_j/d_j$.

In Lemma 5.5 we show that the modified problem can be solved optimally in polynomial time by a greedy algorithm ALG*, which is described in Figure 5.1.

1. Sort bins non-increasingly by efficiency and assume $e_1 \geq \cdots \geq e_m$.
2. Set $x_{t,j} := 0$ for $t = 1, \ldots, n$ and $j = 1, \ldots, m$.
3. Let $x_t = \sum_{j=1}^{m} x_{t,j}$.
4. For $j = 1, \ldots, m$ do
   While bin M_j is not covered do
   (a) If $\nexists J_t \in \sigma : p_t \leq d_j \wedge x_t < p_t$, then stop execution of this while loop, and proceed with the next bin M_{j+1}.
   (b) Otherwise choose an item J_t with largest $p_t \leq d_j$ and $x_t < p_t$.
   (c) If $\sum_{l=1}^{n} x_{l,j} + (p_t - x_t) \leq d_j$, then $x_{t,j} := p_t - x_t$.
   (d) Else $x_{t,j} := d_j - \sum_{l=1}^{n} x_{l,j}$.
5. Output the solution S induced by the $x_{t,j}$ variables.

Figure 5.1: Algorithm ALG*.

Algorithm ALG* considers bins in non-increasing order of efficiency. For each bin M_j algorithm ALG* considers the largest item J_t that is admissible to M_j. If J_t has not yet been assigned or only some parts of J_t have been assigned to some bin previously, then J_t or the respective remaining part of J_t is assigned to M_j. In Step 4c of algorithm ALG* the entire remaining part of item J_t is assigned to the considered bin M_j. Once a bin is covered, the item that exceeds the bin is split

so that the bin is exactly covered, which is done in Step 4d. Then ALG* proceeds with the next smaller item. If there are no items left to assign to the currently considered bin M_j because either M_j is already covered or because there are no items left that are admissible to M_j, then ALG* considers the next bin. We describe below how the solution to be output in Step 5 is determined.

Note that it can happen that ALG* assigns items to bins but these are not covered. Nevertheless, by the definition of the modified problem, they proportionally contribute to the objective function. A solution found by this algorithm is optimal w. r. t. the modified problem, which we show by transforming an optimal solution to a linear program formulation of this modified problem into the solution of ALG* without losing any revenue.

We now give this linear program formulation, which will simplify the transcription of the upcoming analysis of ALG*. Let (μ, σ) be an instance with $\mu = \{M_1, \ldots, M_m\}$ and $\sigma = \{J_1, \ldots, J_n\}$. The modified problem on this instance is described by LP (5.1).

$$\begin{aligned}
\text{maximize} \quad & \sum_{j=1}^{m} r_j y_j && && (5.1)\\
\text{subject to} \quad & y_j \leq \sum_{t=1}^{n} x_{t,j}/d_j && 1 \leq j \leq m\\
& \sum_{j=1}^{m} x_{t,j} \leq p_t && \forall 1 \leq t \leq n\\
& 0 \leq y_j \leq 1 && 1 \leq j \leq m\\
& 0 \leq x_{t,j} && 1 \leq j \leq m, 1 \leq t \leq n\\
& 0 \geq x_{t,j} && \text{for } 1 \leq j \leq m, 1 \leq t \leq n, \text{ where } p_t > d_j.
\end{aligned}$$

Note that we do not need to solve the linear program with one of the known generic methods, for example [46]. In fact, our algorithm ALG* solves this program in time $O(mn)$. In contrast to an arbitrary optimal solution to this linear program, the solution found by ALG* has additional structural properties that we exploit crucially to find a good approximate solution on the same input for GENERALIZED BIN COVERING.

We may identify the non-zero $x_{t,j}$ values in LP (5.1) with item parts $J_{t,l}$ of an item J_t as follows. Assume that $x_{t,j_1}, \ldots, x_{t,j_k} > 0$ are all non-zero variables for a fixed t. Then item J_t consists of parts $J_{t,1}, \ldots, J_{t,k}$ with $p_{t,l} = x_{t,j_l}$ and item part $J_{t,l}$ is assigned to bin M_{j_l}, for $1 \leq l \leq k$. We remark that the variables $x_{t,j}$ in the formal description of Algorithm ALG* in Figure 5.1 coincide with the variables $x_{t,j}$ in LP (5.1). The solution S that is output in Step 5 is created by identifying the non-zero variables $x_{t,j}$ with the item parts of size $p_{t,l}$ of item J_t as just done

above.

Denote with $\text{ALG}^*(\mu, \sigma)$ as usual the value of the solution found by ALG* for the modified problem on the instance (μ, σ). Analogously denote by $\text{OPT}^*(\mu, \sigma)$ the value of an optimal solution to the modified BIN COVERING problem.

Lemma 5.5. *Algorithm* ALG* *gives a solution of value* $\text{ALG}^*(\mu, \sigma) = \text{OPT}^*(\mu, \sigma)$.

Proof. We show that our algorithm gives an optimal solution to LP (5.1) by transforming an arbitrary optimal solution to LP (5.1) into a solution found by our algorithm without losing any revenue. Let $O = (O_1, \ldots, O_m)$ be an optimal solution and assume $(y'_j, x'_{t,j})_{1 \le j \le m, 1 \le t \le n}$ are the corresponding variables describing the assignment of the item parts to bins in O. Let S be the solution found by ALG* and let $(y_j, x_{t,j})_{1 \le j \le m, 1 \le t \le n}$ be the corresponding variables set by ALG*. In iteration $j = 1, \ldots, m$ we set $y'_j := y_j$ and $x'_{t,j} := x_{t,j}$ for all $1 \le t \le n$ and show that the optimality is preserved.

Since items are arbitrarily splittable and by the assumption that all items are assigned by ALG* and OPT, we can assume that $\ell(S_j), \ell(O_j) \le d_j$. Consider the items on bin M_j in the optimal solution and assume that we have already $x'_{t,j'} = x_{t,j'}$ for all $j' < j$ and all $1 \le t \le n$, i. e. bins $M_1, \ldots, M_{j-1}$ in the optimal solution contain only the items or item parts that are also assigned to these bins in the solution S of ALG*.

In case $\ell(S_j) < d_j$ all items being admissible to bin M_j were assigned to the bins $M_1, \ldots, M_j$ in solution S, by construction of the algorithm ALG*. Thus $x_{t,j'} = 0$ for all $1 \le t \le n$ with $p_t \le d_j$ and all $j' > j$. Since there holds $x'_{t,j'} = x_{t,j'}$ for all $j' < j$ and $1 \le t \le n$, it follows that $\sum_{t=1}^{n} x'_{t,j} \le \sum_{t=1}^{n} x_{t,j}$. If even $\sum_{t=1}^{n} x'_{t,j} < \sum_{t=1}^{n} x_{t,j}$, then, in the optimal solution O, the "missing" item parts must be assigned to bins $M_{j'}$ with $j' > j$. And so, if $x'_{t,j} < x_{t,j}$ for some t, we set $x'_{t,j} := x_{t,j}$ and $x'_{t,j'} := 0$ for all $j' > j$. We increase variable y_j and decrease the variables $y_{j'}$, for $j' > j$, corresponding to the changes. By this process the objective value can only be increased since $e_j \ge e_{j'}$ for all $j' > j$ and the sum $\sum_{j=1}^{m} x'_{t,j}$ maintains the same. Also note that no constraints of type $x_{t,j} \le 0$ are violated since ALG* assigns items only to bins to which they are admissible.

Consider the case $\ell(S_j) = d_j$. If $\ell(O_j) < d_j$, then, again because we have already $x'_{t,j'} = x_{t,j'}$ for all $j' < j$ and all $1 \le t \le n$, the "missing" item parts have to reside on bins $M_{j'}$ with $j' > j$, and we can increase some $x'_{t,j}$ variables such that $\sum_{t=1}^{n} x'_{t,j} = \sum_{t=1}^{n} x_{t,j} = \ell(S_t) = d_j$ holds. Naturally, the corresponding variables $x'_{t,j'}$, with $j' > j$, have to be decreased such that the sum $\sum_{j=1}^{m} x'_{t,j}$, for each $1 \le t \le n$, remains unchanged in total and the condition $x_{t,j} \ge 0$ remains satisfied. Also the corresponding y'_j variables have to be adapted. Observe that this can again only increase the objective value. Now, if $x'_{t,j} \ne x_{t,j}$ for some $1 \le t \le n$, then there have to be items $J_t, J_{t'}$ such that for the corresponding

variables there holds $x_{t,j} > x'_{t,j}$ and $x_{t',j} < x'_{t',j}$. For the transformation we proceed in the ordering in which ALG* assigned the items to M_j. Recall that it has done this in non-increasing order of size of the admissible items. Let $J_{t_1}, \ldots, J_{t_k}$ be the items (or item parts) assigned to bin M_j by ALG* in this ordering, i. e. ALG* changed the values of the variables $x_{t_1,j}, \ldots, x_{t_k,j}$ in this ordering.

Under the assumption $x_{t_k,j} > 0$ there holds $\sum_{l=1}^{k-1} x_{t_l,j} < d_j$ since otherwise item J_{t_k} would not have been assigned to bin M_j by ALG*. We conclude that ALG* has assigned the entire yet unassigned parts of each item $J_{t_1}, \ldots, J_{t_k-1}$ to bin M_j in Step 4c. Thus $x_{t_1,j'} = \cdots = x_{t_k-1,j'} = 0$ for $j' > j$. As already $x'_{t,j'} = x_{t,j'}$, for all $1 \leq t \leq n$ and $j' < j$, there holds $x'_{t_l,j} \leq x_{t_l,j}$, for $1 \leq l < k$.

We firstly increase the variables $x'_{t_l,j}$ with $1 \leq l < k$ such that we have $x_{t_l,j} = x'_{t_l,j}$ for all $1 \leq l < k$. Assume $x'_{t_l,j} < x_{t_l,j}$ for some $1 \leq l < k$. For ease of notation set $t^* := t_l$. As $\sum_{t=1}^{n} x'_{t,j} = \sum_{t=1}^{n} x_{t,j} = \ell(S_j) = d_j$, it follows there is some t' such that $x'_{t',j} > x_{t',j}$ and $t' \neq t_l$ for all $1 \leq l < k$ since $x'_{t_l,j} \leq x_{t_l,j}$ for all $1 \leq l < k$.

As $x_{t,j'} = x'_{t,j'}$ for all $1 \leq t \leq n$ and all $1 \leq j' < j$, $x'_{t^*,j} < x_{t^*,j}$ and by the assumption that all items are assigned by OPT, it follows, there is an $j^* > j$ such that $x'_{t^*,j^*} > 0$. We set $\delta := \min\{x'_{t^*,j^*}, x'_{t',j}\}$. Because ALG* assigns admissible items to bins in non-increasing order of size, there holds $p_{t^*} \geq p_{t'}$. Because of $x'_{t^*,j^*} > 0$ item J_{t^*} is admissible to bin M_{j^*} and so is $J_{t'}$. Hence we may set $x'_{t^*,j} := x'_{t^*,j} + \delta$, $x'_{t^*,j^*} := x'_{t^*,j^*} - \delta$, $x'_{t',j} := x'_{t',j} - \delta$ and $x'_{t',j^*} := x'_{t',j^*} + \delta$, and the solution (y', x') remains feasible. After this adjusting there holds $x'_{t^*,j^*} = 0$ or $x'_{t',j} = 0$. As there are only finitely many variables $x_{t^*,j'} > 0$ with $j' > j$ and only finitely many variables $x'_{t',j} > 0$, it follows we have eventually $x_{t_l,j} = x'_{t_l,j}$ for all $1 \leq l < k$ after iteratively adjusting the variables $x'_{t_l,j}$ for $1 \leq l < k$.

If $x_{t_l,j} = x'_{t_l,j}$ for all $1 \leq l < k$, then it also follows $x'_{t_k} \leq x_{t_k}$ because of $x_{t,j'} = x'_{t,j'}$ for all $1 \leq t \leq n$ and all $1 \leq j' < j$ and because of $\sum_{t=1}^{n} x'_{t,j} = \sum_{t=1}^{n} x_{t,j} = \ell(S_j) = d_j$. Then $x'_{t_k,j}$ can be adjusted with the same arguments as above such that $x'_{t_k,j} = x_{t_k,j}$. Observe that the objective function remains unchanged during this adjusting, and we have shown the claim. □

Lemma 5.6. *Let $S = (S_1, \ldots, S_m)$ and $S^* = (S_1^*, \ldots, S_m^*)$ be solutions with the property $\ell(S_j) \leq d_j$ and $\ell(S_j^*) \leq 2d_j$ for all $1 \leq j \leq m$. If for all item parts $J_{t,l}$, with $J_{t,l} \in S_j$ and $J_{t,l} \in S_{j'}^*$, there holds $e_j \leq e_{j'}$, then $r^*(S) \leq 2r^*(S^*)$.*

Proof. If $\ell(S_j) \leq d_j$ for all $1 \leq j \leq m$, then we can compute $r^*(S)$ "itemwise"

$$\begin{aligned} r^*(S) &= \sum_{1 \leq j \leq m} \ell(S_j) e_j = \sum_{1 \leq j \leq m} \sum_{t : J_t \in S_j} p_t e_j \\ &\leq \sum_{1 \leq j \leq m} \sum_{t \in S_j^*} p_t e_j = \sum_{1 \leq j \leq m} \ell(S_j^*) e_j \end{aligned} \tag{5.2}$$

$$\leq \sum_{1 \leq j \leq m} 2 \min\{\ell(S_j^*)/d_j, 1\} d_j e_j \tag{5.3}$$
$$= 2 \sum_{1 \leq j \leq m} r^*(S_j^*) = 2r^*(S^*),$$

where Inequality (5.2) is by $e_j \leq e_{j'}$ for $j > j'$ and the fact that items are assigned in S^* to bins with indices that are not larger than the indices of the bins on which they reside in solution S, by prerequisite. Inequality (5.3) holds since by precondition we have that $\ell(S_j^*) \leq 2d_j$ for each $1 \leq j \leq m$ in S^*. Hence the claim follows. □

A solution of ALG* can be transformed via two steps into a good solution for the GENERALIZED BIN COVERING problem. In Lemma 5.7 we do the first step. By the way ALG* splits items we are able to "reassemble" the split items without losing too much revenue. In the same lemma the solution is then further modified in a greedy way such that there are no items on a not yet covered bin M_j that are admissible to another not yet covered bin $M_{j'}$ with larger efficiency. A solution with this property is called maximal with respect to the modified BIN COVERING problem.

Formally, we call a solution S *maximal with respect to the modified* BIN COVERING *problem* if there are no two distinct bins M_j and $M_{j'}$, with $0 < \ell(S_j) < d_j$, $0 < \ell(S_{j'}) < d_{j'}$ and $e_j \leq e_{j'}$, such that there is an item $J_t \in S_j$ being admissible to bin $M_{j'}$. Note that the maximality of a solution implies the following. If we assign in a maximal solution only one item J_t from bin M_j to bin $M_{j'}$, where for the bins M_j and $M_{j'}$ we have the properties $0 < \ell(S_j) < d_j$, $0 < \ell(S_{j'}) < d_{j'}$ and $e_j \leq e_{j'}$ as above, then bin $M_{j'}$ is already covered by only this single item. This comes from the fact that J_t is not admissible to $M_{j'}$ by the maximality of the solution. We say that a solution S *contains no split items* if for all M_j, $1 \leq j \leq m$, and all $J_t \in S_j$ there holds $p_{t,1} = p_t$.

Lemma 5.7. *Let S be a solution given by* ALG* *for the modified problem. Solution S can be transformed into a solution S^* containing no split items and being maximal with respect to the modified* BIN COVERING *problem such that $r^*(S) \leq 2r^*(S^*)$.*

Proof. In a first step we create a solution S' that contains no split items, i. e. $p_{t,1} = p_t$ for all $1 \leq t \leq n$. Let J_t be an item that is split by ALG* into $n_t > 1$ parts $J_{t,1}, \ldots J_{t,n_t}$. We "merge" the parts $J_{t,1}, \ldots, J_{t,n_t}$ in the following way. We assign all the parts $J_{t,2}, \ldots, J_{t,n_t}$ to the bin to which $J_{t,1}$ was assigned. Refer to the solution created as the solution S'.

By this procedure each bin has received parts of at most one item in solution S': By the algorithm an item J_t is split only when it fills a bin. The bin filled

is assigned a part $J_{t,l}$ and never receives an item or an item part after that again. Hence in solution S each bin M_j contains at most one item that is split for the first time, i. e. if $J_{t,1} \in S_j$ and $J_{t,2} \in S_{j'}$ for some $j' > j$, then there is no item $J_{t'} \neq J_t$ that is split into $n_{t'} > 1$ parts and $J_{t',1} \in S_j$. It follows that in solution S' for each bin M_j there holds $S'_j \setminus S_j = \emptyset$ or $S'_j \setminus S_j = \{J_{l,2}, \ldots, J_{l,n_l}\}$ for some l, i. e. each bin M_j receives in S'_j in addition to the items from S_j only the parts of at most one item.

Assume item J_t was split into $n_t > 1$ parts $J_{t,1}, \ldots, J_{t,n_t}$ by ALG*. If a part $J_{t,1}$ was assigned to a bin M_j, then part $J_{t,1}$ was admissible to M_j and so was the whole item J_t. Thus $p_t \leq d_j$. It follows that $p_{t,1} + \cdots + p_{t,n_t} \leq d_j$. As, by the above argumentation, each bin M_j receives only parts of one item J_t and we had $\ell(S_j) \leq d_j$, by the algorithm ALG*, it follows $\ell(S'_j) \leq \ell(S_j) - p_{t,1} + p_{t,1} + p_{t,2} + \cdots + p_{t,n_t} < 2d_j$, because $p_{t,1} > 0$. Hence we have shown that $\ell(S'_j) < 2d_j$ for each bin M_j, $1 \leq j \leq m$, in solution S'.

As already mentioned, when ALG* splits an item J_t into $n_t > 1$ parts, then the first part $J_{t,1}$ fills a bin. Since ALG* considers the items in non-increasing order of efficiency, we have that the parts $J_{t,2}, \ldots, J_{t,n_t}$ are assigned to bins with at most the same efficiency. Hence in solution S' each item is assigned to a bin M_j with at least the same efficiency as the bin $M_{j'}$ to which it was assigned in solution S.

Solution S' can now be transformed into the maximal solution S^* preserving the both mentioned properties, which are the preconditions of Lemma 5.6. For this let $\mu^* := \{M_j \in \mu \mid 0 < \ell(S'_j) < d_j\}$. Assume w. l. o. g. that $\mu^* = \{M_1, \ldots, M_l\}$ and assume as usual $e_1 \geq \cdots \geq e_l$. For $j = 1, \ldots, l$ do the following. While bin M_j is not covered and one of the bins $M_{j+1}, \ldots, M_l$ contains an item J_t that is admissible to M_j, assign J_t to M_j. If M_j is covered or there are no items left on the bins $M_{j+1}, \ldots, M_l$ that are admissible to M_j, then proceed with bin M_{j+1} and so on. The so created solution is the solution S^*.

Clearly, items are only assigned to more efficient bins. Further, an item J_t is only assigned to a bin M_j when J_t is admissible to M_j and M_j is not yet covered. Hence there holds $\ell(S^*_j) \leq 2d_j$ during and after the entire process. Moreover, if $\ell(S^*_j) < d_j$ for a bin M_j, then by construction all bins $M_{j'} \in \mu^*$ with $e_{j'} \leq e_j$ do not contain any item $J_{t'}$ that is admissible to M_j. Hence S^* is also maximal with respect to the modified BIN COVERING problem. Applying Lemma 5.6 to the solutions S and S^* gives the claim of the lemma. □

From a maximal solution we can create a solution for the GENERALIZED BIN COVERING problem, again by losing only a bounded amount of revenue. For this we move items successively from a not covered bin to the next not covered bin having at least the same efficiency. Since the solution was maximal, the bins with higher efficiency are covered. By this procedure all bins that were assigned items though not covered in the maximal solution are covered now, except for the least

efficient one of these. Either this least efficient bin or the remaining ones yield at least half of the revenue of the bins that were assigned some items in the maximal solution, and we output the one that gains more revenue. Therefore, after this rounding step at most half of the revenue is lost in comparison to the maximal solution. But now, all bins that receive items are actually covered.

Lemma 5.8. *Let S be a solution containing no split items and being maximal with respect to the modified* BIN COVERING *problem. Solution S can be transformed into a solution S^* for the* GENERALIZED BIN COVERING *problem such that $r^*(S) \leq 2r(S^*)$.*

Proof. Let $\mu^* := \{M_j \in \mu \mid 0 < \ell(S_j) < d_j\}$. Assume w. l. o. g. that $\mu^* = \{M_1, \ldots, M_l\}$ and $e_1 \geq \cdots \geq e_l$. Construct two solutions S' and S''. We set $S'_l = \sigma$ and $S'_j = \emptyset$ for $1 \leq j \leq m, j \neq l$. Further we set $S''_{j-1} = S_j$ for $2 \leq j \leq l$, $S''_l = \emptyset$ and $S''_j := S_j$ for $l < j \leq m$.

In S' the only bin M_l that is assigned items is covered since we may assume w. l. o. g. that any bin M_j is covered when all items are assigned to it. Also, all bins in S'' that have received items are covered: Each bin from $\{M_{l+1}, \ldots, M_m\}$ that contains items is covered since it was already covered in S. The bins $M_1, \ldots, M_{l-1}$ are covered since S is a maximal solution with respect to the modified BIN COVERING problem and $e_j \geq e_{j+1}$ holds for $1 \leq j \leq l-1$. Finally, there holds $S''_l = \emptyset$.

As none of the solutions S' and S'' contains split items and all bins that have received items are covered, we have that $r^*(S') = r(S')$ and $r^*(S'') = r(S'')$. We output $S^* := S'$ if $r(S') = \max\{r(S'), r(S'')\}$ and $S^* := S''$ otherwise.

To see the claim about the approximation guarantee distinguish the cases $r_l > r^*(S)/2$ and $r_l \leq r^*(S)/2$, where the index l is as above the index of the bin M_l with $S'_l = \sigma$. If $r_l > r^*(S)/2$, then $r(S^*) \geq r(S') = r_l > r^*(S)/2$. If $r_l \leq r^*(S)/2$, then $r(S^*) \geq r(S'') = r^*(S) - r_l \geq r^*(S) - r^*(S)/2 = r^*(S)/2$, which concludes the proof of the lemma. □

Proof. (of Theorem 5.3). Let (μ, σ) be the given instance. Our algorithm works as follows. We use Observation 5.4 to find a solution $\hat{S}$. Then we run ALG* on the instance (μ, σ) and let S be the solution output. We transform the solution S into a solution S' as done in Lemma 5.7 and then solution S' into solution $\tilde{S}$ as done in Lemma 5.8. We output the better solution from $\{\hat{S}, \tilde{S}\}$. The running time is dominated by the algorithm for MAXIMUM WEIGHT BIPARTITE MATCHING [39].

For the proof of the approximation guarantee, fix an optimal solution O to the instance (μ, σ). Let $\mu_R \subseteq \mu$ be the set of bins covered regularly in solution O and $\sigma_R = \{J_t \in \sigma \mid \exists M_j \in \mu_R : J_t \in O_j\}$, the set of items on these bins. Let $\mu_S \subseteq \mu$ be the set of bins covered singularly by the solution O and $\sigma_S = \{J_t \in \sigma \mid \exists M_j \in \mu_S : J_t \in O_j\}$, the set of items on these bins. We have

$\text{OPT}(\mu,\sigma) = \text{OPT}(\mu_R,\sigma_R) + \text{OPT}(\mu_S,\sigma_S)$. Thus $\text{OPT}(\mu,\sigma) - \text{OPT}(\mu_R,\sigma_R) = \text{OPT}(\mu_S,\sigma_S)$.

If $\text{OPT}(\mu_R,\sigma_R) < 4/5 \cdot \text{OPT}(\mu,\sigma)$, then $\text{OPT}(\mu_S,\sigma_S) > 1/5 \cdot \text{OPT}(\mu,\sigma)$ by the above. Hence in this case $\text{OPT}(\mu,\sigma) \leq 5 \cdot \text{ALG}(\mu,\sigma)$ using Observation 5.4.

Otherwise, if $\text{OPT}(\mu_R,\sigma_R) \geq 4/5 \cdot \text{OPT}(\mu,\sigma)$, then we have $\text{OPT}(\mu_S,\sigma_S) \leq 1/5 \cdot \text{OPT}(\mu,\sigma)$. We find the claimed $\text{OPT}(\mu,\sigma) \leq 5 \cdot \text{ALG}(\mu,\sigma)$ as follows. There holds

$$\begin{aligned}
\text{OPT}(\mu,\sigma) &= \text{OPT}(\mu_R,\sigma_R) + \text{OPT}(\mu_S,\sigma_S) \\
&\leq \text{OPT}^*(\mu_R,\sigma_R) + 1/5 \cdot \text{OPT}(\mu,\sigma) && (5.4)\\
&\leq \text{OPT}^*(\mu,\sigma) + 1/5 \cdot \text{OPT}(\mu,\sigma) \\
&= \text{ALG}^*(\mu,\sigma) + 1/5 \cdot \text{OPT}(\mu,\sigma) && (5.5)\\
&\leq 4 \cdot \text{ALG}(\mu,\sigma) + 1/5 \cdot \text{OPT}(\mu,\sigma), && (5.6)
\end{aligned}$$

where the (in-)equalities are justified as follows. In Inequality (5.4) we use the assumption of the case, $\text{OPT}(\mu_S,\sigma_S) \leq 1/5 \cdot \text{OPT}(\mu,\sigma)$, and $\text{OPT}(\mu_R,\sigma_R) \leq \text{OPT}^*(\mu_R,\sigma_R)$, which clearly holds. Equality (5.5) is provided by Lemma 5.5, and in Inequality (5.6) we account for transforming the fractional solution to the modified problem into a solution for the GENERALIZED BIN COVERING problem with Lemmas 5.7 and 5.8. Finally, $\text{OPT}(\mu,\sigma) \leq 4 \cdot \text{ALG}(\mu,\sigma) + 1/5 \cdot \text{OPT}(\mu,\sigma)$ is equivalent to $\text{OPT}(\mu,\sigma) \leq 5 \cdot \text{ALG}(\mu,\sigma)$, which is the claim.

□

5.3 Variable-Sized Bin Covering in the Unit Supply Model

Recall that in VARIABLE-SIZED BIN COVERING we have $d_j = r_j$ for all $1 \leq j \leq m$, and this is assumed throughout Sections 5.3 and 5.4. For ease of notation, we thus refer with d_j instead of r_j to the revenue of bin M_j. In this subsection we consider the unit supply model.

5.3.1 Tight Analysis of NFD in the Unit Supply Model

The algorithm NEXT FIT DECREASING (NFD) works as follows. Algorithm NFD considers bins in non-increasing order of demand. For each bin, if the total size of the yet unassigned items suffices for coverage, then it assigns items non-increasingly in size until the bin is covered. If the total size of the yet unassigned items does not suffice for coverage, then the bin is skipped. A formal description is given in Figure 5.2. We assume $d_1 \geq \cdots \geq d_m$ and $p_1 \geq \cdots \geq p_n$, as

needed by the algorithm throughout the whole subsection. We show that NFD is a 9/4-approximation.

1. Rename the bins such that $d_1 \geq \cdots \geq d_m$.
2. Rename the items such that $p_1 \geq \cdots \geq p_n$.
3. Let j be the index of the currently considered bin, and set initially $j := 1$.
4. Let t be the index of the first unassigned item, and set initially $t := 1$.
5. While $t \leq n$ and $j \leq m$ do
   (a) If $\sum_{l=t}^{n} p_l < d_j$, then set $j := j + 1$ and $S_j := \emptyset$.
   (b) Otherwise
      (i) Let t' be the smallest index with $\sum_{l=t}^{t'} p_l \geq d_j$.
      (ii) Assign items $J_t, \ldots, J_{t'}$ to bin M_j, i. e., set $S_j := \{J_t, \ldots, J_{t'}\}$
      (iii) Set $j := j + 1$ and $t := t' + 1$.
6. Return $S = (S_j)_{1 \leq j \leq m}$.

Figure 5.2: Algorithm NFD.

Example 5.9. *Let $2/3 > \varepsilon > 0$ be arbitrary. The following instance (μ, σ) yields that* NFD *gives an approximation not better than $9/4 - \varepsilon$. Hence* NFD *is at least a $9/4$-approximation. Let $\mu = \{M_1, \ldots, M_4\}$, with $d_1 = 4$, $d_2 = d_3 = d_4 = 3 - 2\varepsilon$, and $\sigma = \{J_1, \ldots, J_6\}$, with $p_1 = p_2 = p_3 = 2 - \varepsilon$ and $p_4 = p_5 = p_6 = 1 - \varepsilon$. Observe that* $\text{NFD}(\mu, \sigma) = 4$ *and* $\text{OPT}(\mu, \sigma) = 9 - 6\varepsilon$.

Proof Techniques We will use three kinds of arguments. The first type is a bound on the total size of the (remaining) items in the instance and uses no structural properties of the instance. If p^+ is the sum of item sizes in the (remaining) instance, then there holds $\text{OPT} \leq p^+$ since we are considering VARIABLE-SIZED BIN COVERING. Such bounds on the total size of the items are too weak in general to achieve the claimed bound; thus we need arguments actually using the structure of bins in the instance. These are the second type of arguments. For example, if the sum of item sizes in the instance is αd, $\alpha > 1$, and the demand of the only bin in the instance is d, then it follows $\text{OPT} \leq d$ while we could only conclude $\text{OPT} \leq \alpha d$ using a bound on the total size of items. Finally, the third type of arguments allows that we may restrict to instances having certain properties. For

example, we may assume that there are no items in the instance with a size larger than the largest bin demand.

Proof Outline Our proof looks at the specific structure of the solution given by NFD and argues based on that, how much better an optimal solution can be. We employ the described techniques in the following way. Firstly, we settle several basic properties of NFD that will be used implicitly during the analysis. For example, we may assume that NFD covers the first bin (Observation 5.14), and that the bin with smallest demand is empty (Observation 5.15). We will show that we may assume that the "largest" bins are only assigned items such that they do not exceed twice their demand (Lemma 5.16). Here, "largest" bins refers to all bins beginning with the largest bin up to the largest empty bin.

With these tools at hand we can analyze the structure of a solution of NFD. The central notion here is the *well-covered* bin (Definition 5.17). Consider the smallest empty bin in the instance with the property that all bins with larger demand are assigned items with a total size of no more than twice their demand. If such a bin exists, then we call the covered bins being larger than this empty bin well-covered. The proof will be inductive. The terminating cases are the ones when there are either at least four well-covered bins (Observation 5.19) or between two and three well-covered bins but there is a bin among these containing at least three items (Lemma 5.25). These cases are settled by bounds on the total item size, which is the reason why they are terminating cases – even if there are additional filled but not well-covered bins in the instance. We are also in terminating cases if the above prerequisites are not met but when there are no filled bins that are not well-covered. Lemma 5.20 treats the case that all of the at most three well-covered bins contain at most two items and Lemma 5.21 gives the cases in which we have exactly one well-covered bin in the instance.

If there are additional filled but not well-covered bins and applying arguments on the bounds of item sizes is not enough – as in the both last mentioned situations – , we have to look at the instance more closely. Our idea is here to consider a specific not well-covered bin that we call the *head of the instance*. We will subdivide an instance into two parts, which is done by the key lemma of the recursion step, the Decomposition Lemma 5.26. Therein and in Lemma 5.23 we show that it is not advantageous to assign items that NFD has assigned to bins with demand at least d_{j^*} to bins with demand smaller than d_{j^*}, where d_{j^*} is the demand of the head of the instance. This allows us, in combination with some estimations, to split the instance into two parts, which we may investigate separately. The one part containing all bins with demand at least d_{j^*} and all items is analyzed using Lemma 5.24 and Lemma 5.26, which give that the approximation factor of NFD is at most $9/4$ on this subinstance. The other part is a strictly smaller instance and

we may hence iteratively apply the argumentation. We obtain the following result.

Theorem 5.10. *Algorithm* NFD *is a* $9/4$*-approximation, which has running time* $O(n \log n + m \log m)$. *The bound on the approximation factor is tight.*

Note that this is close to being best possible since the problem is inapproximable up to a factor of two, unless P = NP, as shown in Theorem 5.28. We remark that we will speak sometimes slightly imprecisely of a property of an instance (μ, σ) in the following when we actually mean a property of a solution S to the instance (μ, σ). It will be clear from context which solution is meant.

We now start with the proof and settle firstly some basic observations, which are easy but also rather important and will be used often – sometimes implicitly – during the analysis. The first observation is immediate from NFD's behavior, and thus no proof is necessary.

Observation 5.11. *Fix an instance* (μ, σ). *Then the solution of* NFD *(up to renaming items of identical size) for this instance is unique. Furthermore, if* NFD *did not assign item* J_{t^*} *to a bin, then* NFD *does not assign the items* $J_{t^*+1}, \ldots, J_n$ *to a bin either. If a bin* M_j *contains an item* J_t, *then* M_j *is covered.*

Let $O = (O_1, \ldots, O_m)$ be an optimal solution to an instance (μ, σ) and let $S = (S_1, \ldots, S_m)$ be the solution of NFD to the same instance. In the following, we may assume similarly as in Observation 5.11 that if $\ell(O_j) > 0$, then also $\ell(O_j) \geq d_j$. The next observation gives that NFD does not "waste much demand" if many items are assigned to a bin.

Observation 5.12. *Fix a solution* S *of* NFD *to an instance* (μ, σ). *Let* M_j *be a bin containing* $t^* \geq 2$ *items in* S. *Then* $\ell(S_j) < t^*/(t^* - 1)d_j$.

Proof. Assume w. l. o. g. that $S_j = \{J_1, \ldots, J_{t^*}\}$. Since items are ordered non-increasingly by size, it follows $p_{t'} \leq p_t$ for every $1 \leq t \leq t' \leq n$. There holds $p_{t^*-1} < d_j/(t^* - 1)$, which we show by contradiction. If $p_{t^*-1} \geq d_j/(t^* - 1)$, then $p_t \geq d_j/(t^* - 1)$ for every $1 \leq t \leq t^* - 1$. But then $p_1 + \cdots + p_{t^*-1} \geq d_j$. This yields a contradiction to the fact that J_{t^*} is assigned to bin M_j. As $p_{t^*} \leq p_{t^*-1} < d_j/(t^* - 1)$, it follows $\ell(S_j) = \sum_{t=1}^{t^*} p_t < t^*/(t^* - 1)d_j$ because of $\sum_{t=1}^{t^*-1} p_t < d_j$. □

The next observation gives that if a bin M_j in NFD's solution is assigned items with a total size of at least twice d_j, then there is only one item on M_j.

Observation 5.13. *Let* $S = (S_1, \ldots, S_m)$ *be a solution of* NFD *to an instance* (μ, σ). *If* $\ell(S_j) \geq 2d_j$, *then* $|S_j| = 1$.

Proof. This follows immediately by Observation 5.12. □

Note that with the previous observations we also have shown that $\ell(S_j) - p_t < d_j$ for every $J_t \in S_j$. Intuitively, this means that NFD does not assign any items to a bin M_j that are not needed to cover M_j. Similarly, we may assume that the same property holds true in an optimal solution O. The next observation gives that we may restrict ourselves to the analysis of such an instance in which NFD covers the first bin.

Observation 5.14. *Let S be the solution of* NFD *to an instance (μ, σ) and let O be an optimal solution on (μ, σ). If $\ell(S_1) = \cdots = \ell(S_{j^*}) = 0$, then $\ell(O_1) = \cdots = \ell(O_{j^*}) = 0$. Let $\mu'' = \mu \setminus \{M_1, \ldots, M_{j^*}\}$. Then $\text{NFD}(\mu, \sigma) = \text{NFD}(\mu'', \sigma)$ and $\text{OPT}(\mu, \sigma) = \text{OPT}(\mu'', \sigma)$.*

Proof. If $\ell(S_1) = \cdots = \ell(S_{j^*}) = 0$, then $\sum_{t=1}^{n} p_t < d_j$ for all $1 \leq j \leq j^*$ since otherwise NFD would have covered at least one of the bins $M_1, \ldots, M_{j^*}$. Hence, also OPT can not cover any of the bins $M_1, \ldots, M_{j^*}$. □

By the argument given by the previous observation it is also justified to assume $\text{NFD}(\mu, \sigma) > 0$. Since otherwise also $\text{OPT}(\mu, \sigma) = 0$ follows and NFD is optimal. This assumption will always be implicitly used and thus the quotient $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma)$ is always defined. Alternatively, if $\text{NFD}(\mu, \sigma) = 0$, we could define $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) := 1$ since in this case also $\text{OPT}(\mu, \sigma) = 0$. Further, we may always assume that there exists an empty bin, otherwise NFD is clearly optimal. We may strengthen this observation such that it suffices to compare instances of NFD to OPT, where the smallest bin, M_m, is empty.

Observation 5.15. *Fix a solution S of* NFD *to an instance (μ, σ). Let M_{j^*} be a bin with $\ell(S_{j^*}) = 0$, and for all j, with $j^* < j \leq m$, there holds $\ell(S_j) > 0$. Let $\sigma^* \supseteq \sigma$. Then $\text{OPT}(\mu, \sigma^*)/\text{NFD}(\mu, \sigma) \leq \text{OPT}(\mu', \sigma^*)/\text{NFD}(\mu', \sigma)$, where $\mu' = \{M_1, \ldots, M_{j^*}\}$.*

Proof. Let S' be a solution of NFD to the instance (μ', σ), and recall that S is the solution of NFD to the instance (μ, σ). Let t^* be the smallest index of an item J_t such that $J_t \in S_{j^*+1} \cup \cdots \cup S_m$. It is an elementary property of algorithm NFD that it does not assign an item that resides on one bin $M_{j^*+1}, \ldots, M_m$ in solution S to the instance (μ, σ) to one of the bins $M_1, \ldots, M_{j^*}$ in solution S' to the instance (μ', σ). Formally, there is no $J_t \in S_{j^*+1} \cup \cdots \cup S_m$ such that $J_t \in S'_1 \cup \cdots \cup S'_{j^*}$ since otherwise NFD would have assigned item J_t to a bin M_j, with $1 \leq j \leq j^*$, in solution S already. We conclude $S_j = S'_j$ for all $1 \leq j \leq j^*$. Thus it follows

$$\text{NFD}(\mu, \sigma) = \text{NFD}(\mu', \sigma) + \sum_{j=j^*+1}^{m} d_j \tag{5.7}$$

because all bins $M_{j^*+1}, \ldots, M_m$ are covered in S by precondition.

For OPT we have

$$\text{OPT}(\mu', \sigma^*) + \sum_{j=j^*+1}^{m} d_j \geq \text{OPT}(\mu, \sigma^*) \tag{5.8}$$

since an optimal algorithm can possibly cover additional bins from $M_1, \ldots, M_{j^*}$ with items that reside on a bin from $M_{j^*+1}, \ldots, M_m$ in its solution to the instance (μ, σ^*).

Combining (5.7) and (5.8) yields

$$\frac{\text{OPT}(\mu', \sigma^*)}{\text{NFD}(\mu', \sigma)} \geq \frac{\text{OPT}(\mu, \sigma^*) - \sum_{j=j^*+1}^{m} d_j}{\text{NFD}(\mu, \sigma) - \sum_{j=j^*+1}^{m} d_j} \geq \frac{\text{OPT}(\mu, \sigma^*)}{\text{NFD}(\mu, \sigma)},$$

where the last inequality holds because of the relation $\text{OPT}(\mu, \sigma^*) \geq \text{OPT}(\mu, \sigma) \geq \text{NFD}(\mu, \sigma) \geq \sum_{j=j^*+1}^{m} d_j \geq 0$. □

With this observation let $\ell(S_m) = 0$ from now on. Let j^* be the smallest index such that $\ell(S_{j^*+1}) = 0$. The next lemma states that we can assume w. l. o. g. that for all bins M_j, with $1 \leq j \leq j^*$, there holds $\ell(S_j) < 2d_j$.

Lemma 5.16. *Fix a solution S of* NFD *to an instance (μ, σ). Let j^* be the smallest index such that M_{j^*} and M_{j^*+1} are bins with $\ell(S_{j^*}) > 0$ and $\ell(S_{j^*+1}) = 0$. Let $\mu' = \{M_j \in \mu \mid \ell(S_j) < 2d_j \text{ or } j \geq j^*\}$ and $\sigma' = \{J_t \in \sigma \mid \exists M_j \in \mu' : J_t \in S_j\}$. Then $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) \leq \text{OPT}(\mu', \sigma')/\text{NFD}(\mu', \sigma')$.*

Proof. Consider the solution S of NFD. Let $j_1 \leq \cdots \leq j_k$, with $j_k \leq j^*$, be all indices such that $\ell(j_l) \geq 2d_{j_l}$ for $l = 1, \ldots, k$. Let $J_{t_1}, \ldots, J_{t_k}$ be the items in non-increasing order of size on these bins in S, where we have by Observation 5.13 that on each M_{j_l} in fact resides only one item. Consider bin M_{j_1} and item J_{t_1}. We argue that we can assume that an optimal algorithm also assigns J_{t_1} to M_{j_1}.

By the fact that bins are ordered non-increasingly by demand, if we assign J_{t_1} to a bin M_j, with $j \geq j_1$, we have that $p_{t_1} - d_j \geq p_{t_1} - d_{j_1} = \ell(j_1) - d_{j_1}$. Because of this and because every bin M_j, with $j \geq j_1$, is covered only by item J_{t_1}, we can assume an optimal algorithm would assign J_{t_1} to a bin M_j, with $j \leq j_1$.

If, in optimal solution O, J_{t_1} is assigned to a bin with index smaller than t_1, then it can be assumed that not all of the items $J_1, \ldots, J_{t_1-1}$ are assigned to only the bins $M_1, \ldots, M_{j_1-1}$. This is because also NFD assigns the items $J_1, \ldots, J_{t_1-1}$ to the bins $M_1, \ldots, M_{j_1-1}$ and these are already covered. Hence one of the items $J_1, \ldots, J_{t_1}$ could potentially cover an additional bin, and we thus can assume that one of the items $J_1, \ldots, J_{t_1}$ is assigned to a bin M_j, with $j \geq j_1$, in O since this is not worse.

With the same argumentation as above, for every such an item J_t, if it would be assigned to a bin M_j, $j > j_1$, then we had $p_t - d_j \geq p_t - d_{j_1}$. Because every such

an item J_t covers every bin M_j, with $j \geq j_1$, we can thus assume J_t is assigned to M_{j_1}. But then, since $p_t - d_{j_1} \geq p_{t_1} - d_{j_1}$, we can also assume J_{t_1} is assigned in an optimal solution O to M_{j_1}.

Hence, defining $\mu'' = \mu \setminus \{M_{j_1}\}$ and $\sigma'' = \setminus\{J_{t_1}\}$ we have

$$\frac{\text{OPT}(\mu,\sigma)}{\text{NFD}(\mu,\sigma)} = \frac{d_{j_1} + \text{OPT}(\mu'',\sigma'')}{d_{j_1} + \text{NFD}(\mu'',\sigma'')} \leq \frac{\text{OPT}(\mu'',\sigma'')}{\text{NFD}(\mu'',\sigma'')},$$

where the last inequality is because $\text{OPT}(\mu'',\sigma'') \geq \text{NFD}(\mu'',\sigma'') \geq d_{j_1} \geq 0$. Iteratively applying this argumentation yields the claim. □

In order to simplify the analysis we define the notion of a *well-covered bin*.

Definition 5.17. *Consider a solution S of* NFD *to an instance (μ,σ). Fix a bin M_{j^*}, with $\ell(j^*) > 0$, and let j' be the smallest number with $j' > j^*$ such that $\ell(j') = 0$ if it exists. The bin M_{j^*} is said to be well-covered if j' exists and $\ell(j) \leq 2d_j$ for all $j = 1,\ldots,j'$.*

By Observation 5.14 and Lemma 5.16 it can be shown that we may assume that there is at least one well-covered bin in the instance.

Observation 5.18. *Consider a solution S of* NFD *to an instance (μ,σ) that contains at least one filled bin. The number k of well-covered bins is well-defined and we can assume $k \geq 1$.*

Proof. Let j^* be the smallest index such that $\ell(j^*) = 0$. By Observation 5.15 we may assume that bin M_m is empty, and hence $j^* \leq m$ exists. Further we can assume $j^* > 1$ by Observation 5.14. By Lemma 5.16 we may assume that $\ell(j) \leq 2d_j$ for all $j = 1,\ldots,j^*$. Hence a largest index $\hat{j} \geq j^*$ with the property $\ell(j) \leq 2d_j$ for all $j \leq \hat{j}$ exists, and the set $\mu^* = \{M_j \mid j \leq \hat{j}, \ell(j) > 0\}$ is unique. By definition, set μ^* contains all well-covered bins, and as $M_1 \in \mu^*$ because of $\hat{j} > 1$, it also follows $|\mu^*| = k \geq 1$. □

Observation 5.19. *Let (μ,σ) be an instance. If* NFD *gives a solution containing k well-covered bins, then $\text{OPT}(\mu,\sigma)/\text{NFD}(\mu,\sigma) \leq 2 + 1/k$.*

Proof. Let k' be the largest index of a well-covered bin and let μ^* be the set of well-covered bins. On the one hand we have $\text{NFD}(\mu,\sigma) \geq \sum_{j:M_j\in\mu^*} d_j$ and on the other $\text{NFD}(\mu,\sigma) \geq kd_{k'}$. Recall, we have for every $M_j \in \mu^*$ that $\ell(j) \leq 2d_j$. Let t^* be the smallest index of an item that NFD did not assign to a bin from μ^*. Since $\ell(k'+1) = 0$ by choice of k', we conclude $\sum_{t=t^*}^{n} p_t < d_{k'+1} \leq d_{k'}$ as otherwise NFD would have filled bin $M_{k'+1}$. It follows $\sum_{t=1}^{n} p_t < \sum_{j:M_j\in\mu^*} 2d_j + d_{k'}$. Because $\text{OPT}(\mu,\sigma) \leq \sum_{t=1}^{n} p_t$, we can bound $\text{OPT}(\mu,\sigma) < \sum_{j:M_j\in\mu^*} 2d_j + d_{k'} \leq 2\text{NFD}(\mu,\sigma) + 1/k \cdot \text{NFD}(\mu,\sigma) = (2 + 1/k)\text{NFD}(\mu,\sigma)$. □

For a number of $k \geq 4$ well-covered bins we already have the desired result. Also, we can already conclude that NFD is at most a 3-approximation. We now turn our attention to the cases when $k \leq 3$.

Lemma 5.20. *Let (μ, σ) be an instance. If* NFD *gives a solution in which every filled bin is well-covered and contains at most two items, then there holds* $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) \leq 2$.

Proof. Call a maximal set $\{M_{j'}, M_{j'+1}, \ldots, M_{j''}\}$ of bins with subsequent indices with $\ell(S_{j'}) = \cdots = \ell(S_{j''}) = 0$ a gap G_l. Let $G_1, \ldots, G_k$ be all gaps in solution S, and assume that the gaps are enumerated such that for $M_j \in G_i$ and $M_{j'} \in G_{i'}$ with $1 \leq i < i' \leq k$ there holds $j < j'$.

Analogously, enumerate the maximal sets $\{M_{j'}, M_{j'+1}, \ldots, M_{j''}\}$ of bins with subsequent indices with $\ell(S_{j'}) > 0$, for $j' \leq j \leq j''$, and refer to them as the sets $F_1, \ldots, F_{k'}$ of filled bins. By Observation 5.14 we may assume $\ell(S_1) > 0$ and by Lemma 5.16 we may assume $\ell(S_m) = 0$. Hence $k = k'$, and thus $F_1, \ldots, F_k, G_1, \ldots, G_k$ is a partition of the bins in μ.

For each $1 \leq l \leq k$, if there are n_l many items on the bins in F_l in S, then we modify the set G_l in such a way that $|F_l| + |G_l| = n_l$ if not already the case. If $|G_l| < n_l - |F_l|$, then introduce $n_l - |F_l| - |G_l|$ many bins with demand d_{j^*} into G_l, where M_{j^*} is the bin with largest index in G_l. If $|F_l| + |G_l| > n_l$, then remove the $|F_l| + |G_l| - n_l$ bins with smallest demand from G_l. Observe that removing this number of bins is always possible since $|G_l| - (|F_l| + |G_l| - n_l) \geq 0$ because of $|F_l| \leq n_l$. Refer to the modified set G_l as the set G'_l. We remark that it is no problem if there exists a gap G'_l such that $|G'_l| = 0$. Set $\bar{\mu} = \bigcup_{1 \leq l \leq k}(F_l \cup G'_l)$.

Let S be the solution of NFD to the instance (μ, σ) and let S' be the solution of NFD to the instance $(\bar{\mu}, \sigma)$. Observe, we not only have $\text{NFD}(\bar{\mu}, \sigma) = \text{NFD}(\mu, \sigma)$, but also all items are assigned to the same bins in both solutions. Formally, $S_j = S'_j$ for every M_j such that $M_j \in F_l$ for some $1 \leq l \leq k$ and $S'_j = \emptyset$ for all $M_j \in G'_l$ and all $1 \leq l \leq k$. This holds true because we have copied and removed only bins that were not covered in solution S.

Note that, if $M_{j^*} \in G_l$ is an empty bin in solution S, then NFD can not fill a bin $M_{j'} \in F_{l'} \cup G_{l'} \cup G'_{l'}$, with $l' \leq l$, by only the items from $\bigcup_{j:M_j \in F_{l+1} \cup \cdots \cup F_k} S_j$. Then also OPT can only fill such a bin $M_{j'} \in F_{l'} \cup G_{l'} \cup G'_{l'}$ if there is also an item from the set $\bigcup_{j:M_j \in F_1 \cup \cdots \cup F_l} S_j$ on this bin.

Consider the following relaxed BIN COVERING problem. In order to earn the revenue d_j for a bin M_j either bin M_j has to be covered (with some items) or an item J_t from the set $S_1 \cup \cdots \cup S_j$ has to be assigned to M_j. The problem is clearly a relaxation of the ordinary VARIABLE-SIZED BIN COVERING problem. Let OPT^* denote an optimal algorithm for the relaxed problem. As the problem is a relaxation, there holds $\text{OPT}^*(\mu, \sigma) \geq \text{OPT}(\mu, \sigma)$.

We also prove that $\text{OPT}^*(\bar{\mu}, \sigma) \geq \text{OPT}^*(\mu, \sigma)$. In order to show this, we justify that the removal of bins from the instance $(\bar{\mu} \cup \bigcup_{1 \leq l \leq k} G_l, \sigma)$ does not make OPT^* lose any revenue.

Consider an arbitrary bin $M_{j^*} \in F_{l^*} \cup G_{l^*}$, for an $1 \leq l^* \leq k$. Recall that no subset of the items assigned in S to the bins from $F_{l^*+1} \cup \cdots \cup F_k$ can cover bin M_{j^*} without an item residing on a bin from $F_1 \cup \cdots \cup F_{l^*}$ in S. But if any item J_t residing on a bin from $F_1 \cup \cdots \cup F_{l^*}$ in S is assigned to bin M_{j^*}, then OPT^* already gains the revenue for this bin M_{j^*} without assigning another item to this bin, by definition of our relaxation. In conclusion, we can assume OPT^* does not assign an item J_t residing on a bin from $F_{l^*+1} \cup \cdots \cup F_k$ in S to a bin from $F_{l^*} \cup G_{l^*} \cup G'_{l^*}$ in the solution to $(\bar{\mu} \cup \bigcup_{1 \leq l \leq k} G_l, \sigma)$, and hence, in particular, no bin from G_{l^*} contains such an item.

Also observe that assigning an item J_t residing on a bin $F_{l'}$, with $l' < l^*$, in solution S to a bin from $F_{l^*} \cup G_{l^*} \cup G'_{l^*}$ is not advantageous since all bins in $F_{l'} \cup G'_{l'}$ have at least the same demand as the largest bin in $F_{l^*} \cup G_{l^*} \cup G'_{l^*}$ and, by construction, $\bigcup_{1 \leq l \leq l'}(F_l \cup G'_l)$ contains $n_1 + \cdots + n_{l'} = \sum_{j : M_j \in F_1 \cup \cdots \cup F_{l'}} |S_j|$ many bins. Hence, each item $J_t \in S_j$, with $M_j \in F_l$, is assigned to a bin from $F_l \cup G_l \cup G'_l$ and the bins in $G_{l^*} \setminus G'_{l^*}$, if there are any, remain empty. Thus, those bins can be removed. As our argumentation holds for every set $F_{l^*} \cup G_{l^*}$, the claim follows. Notice that it is an interesting property of the solution of OPT^* that in this modified instance each bin contains exactly one of the items that reside on a well-covered bin in NFD's solution.

We now argue that $\text{OPT}^*(\bar{\mu}, \sigma) \leq 2\text{NFD}(\bar{\mu}, \sigma)$. We associate every bin from some gap G'_l to a bin with at least the same demand from F_l, and two distinct bins from G'_l are never associated to the same bin. If n_l is the number of items on the bins from F_l in solution S', then we have $|F_l| + |G'_l| = n_l$, by construction of the instance $\bar{\mu}$. Observe that $|F_l| \geq n_l/2$ as at most two items reside by prerequisite on every bin from F_l. It follows $|F_l| \geq |G'_l|$. By definition of the set G'_l, every bin in G'_l has at most as much demand as the smallest bin from F_l. As we have argued that OPT^* earns the revenue for the bins in $F_l \cup G'_l$ whereas NFD gains the revenue for the bins in F_l, it follows $\text{OPT}^*(\bar{\mu}, \sigma) \leq 2\text{NFD}(\bar{\mu}, \sigma)$.

In conclusion, we have shown the inequality

$$2\text{NFD}(\mu, \sigma) = 2\text{NFD}(\bar{\mu}, \sigma) \geq \text{OPT}^*(\bar{\mu}, \sigma) \geq \text{OPT}^*(\mu, \sigma) \geq \text{OPT}(\mu, \sigma),$$

which yields the claim. □

Lemma 5.21. *Let (μ, σ) be an instance. If* NFD *gives a solution with $k = 1$ well-covered bins and all other bins are empty, then* $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) \leq 9/4$.

Proof. Fix a solution S of NFD and for ease of notation let $t^* = |S_1|$. Recall that by the ordering in which NFD considers the items, it follows $S_1 = \{J_1, \ldots, J_{t^*}\}$.

We may further assume that $t^* \geq 3$ since otherwise the claim follows immediately from Lemma 5.20.

By Observation 5.12 $\sum_{t=1}^{t^*} p_t < \frac{t^*}{t^*-1} d_1$. As $d_m \leq d_1$ and bin M_m is not covered in S, it follows that $\sum_{t=t^*+1}^{n} p_t < d_m \leq d_1$. Altogether we can bound

$$\text{OPT}(\mu, \sigma) \leq \sum_{t=1}^{n} p_t = \sum_{t=1}^{t^*} p_t + \sum_{t=t^*+1}^{n} p_t < \frac{t^*}{t^*-1} d_1 + d_m \leq 3/2 d_1 + d_m, \tag{5.9}$$

where the last inequality holds by the assumption $t^* \geq 3$.

Assume that in an optimal solution the bins $M_{j_1}, \ldots, M_{j_k}$ are covered and that $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) > 9/4$. Because $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) > 9/4$, it follows $k \geq 3$ since $\text{ALG}(\mu, \sigma) = d_1$ and $d_j \leq d_1$ for all $1 \leq j \leq m$.

If it were true that $d_m < 3/4 d_1$, then by Estimation (5.9) it follows that $\text{OPT}(\mu, \sigma) < 3/2 d_1 + 3/4 d_1 = 9/4 d_1$, and since $\text{ALG}(\mu, \sigma) = d_1$, this would yield a contradiction to the assumption $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) > 9/4$. Hence $d_m \geq 3/4 d_1$, and thus also $d_{j_1}, \ldots, d_{j_k} \geq 3/4 d_1$ because bins are ordered non-increasingly by demand.

As the bins $M_{i_1}, \ldots, M_{i_k}$ are covered in an optimal solution, there has to hold

$$\sum_{t=1}^{n} p_t \geq d_{j_1} + \cdots + d_{j_k} \geq 3/2 d_1 + d_m, \tag{5.10}$$

where the last inequality follows because of $k \geq 3$, as shown above, and since $d_{j_k} \geq d_m$. But (5.10) yields a contradiction to (5.9), and we have thus shown that the assumption $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) > 9/4$ is false. This proves the lemma. □

In order to simplify the following statements we introduce the term head of the instance, which is a distinguished bin. For this, fix a solution of NFD to a given instance (μ, σ). Let j_0 be the index of the first not well-covered bin with $\ell(j_0) > 0$ and let j_1 be the smallest index, with $j_1 \geq j_0$, such that $\ell(j_1 + 1) = 0$. Let $j^* = \max_{j:\ell(j)>2d_j}\{j \leq j_1\}$. Then the bin M_{j*} is called the *head* of the instance.

Observation 5.22. *Fix a solution of* NFD *to the instance* (μ, σ)*. If there is a filled not well-covered bin, then there is a unique bin* M_{j^*} *being the head of the instance.*

Proof. Recall that $j^* = \max_{j:\ell(j)>2d_j}\{j \leq j_1\}$. We verify that the set over which the maximum is taken is non-empty. By Observation 5.18 we can assume there exist at least $k \geq 1$ well-covered bins in every instance. Observe by definition that if there is an index j, with $\ell(j) > 0$, such that M_j is not a well-covered bin, then all bins $M_{j'}$, with $\ell(j') > 0$ and $j' > j$, are also not well-covered.

Let j_0 now be the smallest index of a bin M_{j_0}, with $\ell(j_0) > 0$, that is not well-covered. Note that M_{j_0} exists by precondition. As $\ell(m) = 0$ as guaranteed by Observation 5.15, there exists a smallest index j_1, with $j_1 > j_0$, and $\ell(j_1+1) = 0$. By the definition of well-coverage there has to be an index j, with $j_0 \leq j < j_1$, such that $\ell(j) > 2d_j$. Now, let $j^* \leq j_1$ be the largest of such indices. Thus the indices j_0, j_1 and j^* exist and are unique. Hence we have shown that if the solution of NFD contains a not well-covered non-empty bin, then the head of the instance is uniquely determined. □

Lemma 5.23. *Let (μ, σ) be an instance to which* NFD *gives a solution S with $k = 1$ well-covered bin and S further contains an additional filled bin that is not well-covered. Let M_{j^*} be the head of the instance. If there are at least three items on bin M_1 in S and in an optimal solution at least one of these items is assigned to a bin with index at least j^*, then* $\text{OPT}(\mu, \sigma)/\text{NFD}(\mu, \sigma) \leq 9/4$.

Proof. Let $|S_1| = t^*$. Since $t^* \geq 3$, we have that the item J_{t^*} has size $p_{t^*} \leq d_1/2$, by Observation 5.12. Since there is only one well-covered bin in the instance, we find $\ell(S_2) = 0$. Let $J_{t'}$ be the item on bin M_{j^*} and recall that $p_{t'} > 2d_{j^*}$ since M_{j^*} is the head of the instance. For every item J_t from bin M_1 we have that $p_t \geq p_{t'}$. Thus there holds for every such an item $J_t \in S_1$ that it will not only cover a bin M_j, with $j \geq j^*$, but there even holds that

$$d_j \leq d_{j^*} < p_{t'}/2 \leq p_t/2. \tag{5.11}$$

By the ordering of items by size we observe the following. If in an optimal solution at least one of the items from bin M_1 is assigned to a bin M_j, with $j \geq j^*$, then we can assume that – besides possibly other items – also the item J_{t^*} is assigned to such a bin $M_{j'}$, with $j' > j^*$. This can only better than choosing an item with index smaller than t^* by the fact that each such an item will fill its respective bin M_j. For the remaining $t^* - 1$ items on bin M_1 we can bound $\sum_{t=1}^{t^*-1} p_t < d_1$ because bin M_1 was not yet covered when item J_{t^*} was assigned to it by NFD.

As in Lemma 5.21 we can bound $\sum_{t=t^*+1}^{n} p_t < d_1$. Hence, when assigning at least one of the items from S_1 to a bin M_j, with $j \geq j^*$, we can bound the revenue OPT yields with Inequality (5.11) by

$$\text{OPT}(\mu, \sigma) < \sum_{t=1}^{t^*-1} p_t + p_{t^*}/2 + \sum_{t=t^*+1}^{n} p_t < d_1 + (d_1/2)/2 + d_1 = 9/4 d_1,$$

and as $\text{NFD}(\mu, \sigma) \geq d_1$, the claim follows. □

Lemma 5.24. *Fix an instance (μ, σ). Let S be the solution of* NFD *and O an optimal solution to (μ, σ). Assume that S contains $k \in \{1, 2, 3\}$ well-covered*

bins and each of them contains at most two items. Let S further have the property that there is at least one filled bin that is not well-covered, and assume M_{j^*} is the head of the instance. Define $\mu' = \{M_1, \dots, M_{j^*}\}$ and $\mu'' = \mu \setminus \mu'$. Let $\sigma'_1 = S_1 \cup \dots \cup S_{j^*}$ and $\sigma'_2 = O_1 \cup \dots \cup O_{j^*}$. Finally, let $\sigma''_1 = \sigma \setminus \sigma'_1$ and $\sigma''_2 = \sigma \setminus \sigma'_2$. There holds $(\text{OPT}(\mu', \sigma'_2) + \text{OPT}(\mu'', \sigma''_2 \setminus \sigma''_1))/\text{NFD}(\mu', \sigma'_1) \leq 2$.

Proof. We firstly show a property that holds independently of how many well-covered bins an instance contains and independently of how many items on each of these bins reside. Let j' be the smallest index of a bin in NFD's solution that is filled but not well-covered. Note that by the definitions of well-covered bins and the head of the instance that all bins $M_{j'}, \dots, M_{j^*}$ contain items. Let $\mu^* = \{M_1, \dots, M_{j'-1}\}$ and S^* be the set of items residing on the well-covered bins, i. e. $S^* = \bigcup_{1 \leq i \leq j'-1} S_j$. There holds

$$\text{OPT}(\mu'', \sigma''_2 \setminus (S^* \cup \sigma''_1)) \leq \sum_{j' \leq j \leq j^*} d_j. \tag{5.12}$$

We show Property (5.12). There holds

$$\begin{aligned}
\sum_{j' \leq j \leq j^*} d_j &= \sum_{\substack{j' \leq j \leq j^*, \\ |S_j|=1}} d_j + \sum_{\substack{j' \leq j \leq j^*, \\ |S_j| \geq 2}} d_j \\
&\geq \sum_{\substack{j' \leq j \leq j^*, \\ |S_j|=1}} d_{j^*} + \sum_{\substack{j' \leq j \leq j^*, \\ |S_j| \geq 2}} d_j \\
&\geq \sum_{\substack{j' \leq j \leq j^*, \\ |S_j|=1}} d_{j^*} + \sum_{\substack{j' \leq j \leq j^*, \\ |S_j| \geq 2}} \sum_{t: J_t \in S_j} p_t/2 && (5.13) \\
&\geq \sum_{\substack{j' \leq j \leq j^*, \\ |S_j|=1}} d_{j^*} + \sum_{\substack{j' \leq j \leq j^*, \\ |S_j| \geq 2}} \sum_{t: J_t \in S_j} d_{j^*} && (5.14) \\
&= \sum_{j' \leq j \leq j^*} \sum_{J_t \in S_j} d_{j^*} \\
&= d_{j^*} \sum_{j' \leq j \leq j^*} |S_j| \\
&\geq \text{OPT}(\mu'', \sigma''_2 \setminus (S^* \cup \sigma''_1)), && (5.15)
\end{aligned}$$

where the inequalities are justified as follows. In (5.13) we use that if $|S_j| \geq 2$, then $\sum_{t: J_t \in S_j} p_t = \ell(j) < 2d_j$ by Observation 5.12. Since M_{j^*} is the head of the

instance, there holds $p_{t^*} > 2d_{j^*}$ for the only item J_{t^*} that resides on M_{j^*}. Hence $p_t > 2d_{j^*}$ holds for every item $J_t \in S_1 \cup \cdots \cup S_{j^*}$ since items are assigned sorted non-increasingly in size by NFD. This shows Inequality (5.14).

Inequality (5.15) holds since $\sigma_2'' \setminus (S^* \cup \sigma_1'') \subseteq \sigma_1' \setminus S^* = \bigcup_{j' \leq j \leq j^*} S_j$ and because each item $J_t \in \bigcup_{j' \leq j \leq j^*} S_j$ that is assigned to a bin from μ'' covers at most one bin with demand at most d_{j^*}.

We now investigate the case that $k \in \{1, 2, 3\}$ bins are well-covered in NFD's solution and that each of these well-covered bins contains at most two items. Let $\sigma_R = S^* \cap \sigma_2''$ be the subset of the items that reside on a well-covered bin in NFD's solution and that are assigned to a bin from μ'' by the optimal algorithm. We firstly observe that $\text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1') \leq \text{OPT}(\mu'', \sigma_R) + \text{OPT}(\mu'', \sigma_2'' \setminus (\sigma_R \cup \sigma_1'))$. This holds true because the items from σ_R reside alone on a bin from μ'' in an optimal solution because each one covers a bin without any additional items. Recall that all bins $M_{j'}, \ldots, M_{j^*}$ are covered in S. Hence, $\text{NFD}(\mu', \sigma_1') = \text{NFD}(\mu^*, \sigma_1') + \sum_{j' \leq j \leq j^*} d_j$ by definition of sets μ^* and σ_1'.

Consider again the construction from Lemma 5.20 and the optimal algorithm OPT^* for the relaxation defined therein. Note that the input instance (μ^*, σ) fulfills the prerequisites of this lemma. Let $\bar{\mu}$ denote the set of bins of which the modified instance in Lemma 5.20 consists when Lemma 5.20 is applied on the instance (μ^*, σ), i. e. $\bar{\mu}$ is the union of the sets F_l and G_l'. For sake of clarity, let us also emphasize that the sets of bins referred to as μ within this lemma is referred to as μ^* here. With these notions we show that $\text{OPT}(\mu^*, \sigma_2') + \text{OPT}(\mu'', \sigma_R) \leq \text{OPT}^*(\bar{\mu}, \sigma)$. Assuming this holds true, we can show the statement of the lemma as follows.

$$
\begin{aligned}
&\frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'')}{\text{NFD}(\mu', \sigma_1')} \\
&\leq \frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_R) + \text{OPT}(\mu'', \sigma_2'' \setminus (\sigma_R \cup \sigma_1''))}{\text{NFD}(\mu', \sigma_1')} \\
&\leq \frac{\text{OPT}(\mu^*, \sigma_2') + \sum_{j' \leq j \leq j^*} d_j + \text{OPT}(\mu'', \sigma_R) + \text{OPT}(\mu'', \sigma_2'' \setminus (\sigma_R \cup \sigma_1''))}{\text{NFD}(\mu^*, \sigma_1') + \sum_{j' \leq j \leq j^*} d_j} \\
&= \frac{\text{OPT}(\mu^*, \sigma_2') + \text{OPT}(\mu'', \sigma_R) + \sum_{j' \leq j \leq j^*} d_j + \text{OPT}(\mu'', \sigma_2'' \setminus (S^* \cup \sigma_1''))}{\text{NFD}(\mu^*, \sigma) + \sum_{j' \leq j \leq j^*} d_j} \qquad (5.16) \\
&\leq \frac{\text{OPT}^*(\bar{\mu}, \sigma) + \sum_{j' \leq j \leq j^*} d_j + \text{OPT}(\mu'', \sigma_2'' \setminus (S^* \cup \sigma_1''))}{\text{NFD}(\mu^*, \sigma) + \sum_{j' \leq j \leq j^*} d_j}
\end{aligned}
$$

$$\leq \max\left\{\frac{\text{OPT}^*(\bar{\mu},\sigma)}{\text{NFD}(\mu^*,\sigma)}, \frac{\sum_{j'\leq j\leq j^*} d_j + \text{OPT}(\mu'', \sigma_2'' \setminus (S^* \cup \sigma_1''))}{\sum_{j'\leq j\leq j^*} d_j}\right\} \tag{5.17}$$

$$\leq 2, \tag{5.18}$$

where the (in-)equalities are justified as follows. In Equality (5.16) we use in the denominator that NFD assigns no items from σ_1'' to a bin from μ', by definition of σ_1', and in the numerator that σ_R contains all items from S^* that are also in σ_2''. Inequality (5.17) uses elementary calculus. In Inequality (5.18) we bound $\text{OPT}^*(\bar{\mu},\sigma)/\text{NFD}(\mu^*,\sigma) \leq 2$ as this is done in the proof of Lemma 5.20, and finally use Property (5.12) to bound the right hand term in the maximum. We are left to show that $\text{OPT}(\mu^*,\sigma_2') + \text{OPT}(\mu'',\sigma_R) \leq \text{OPT}^*(\bar{\mu},\sigma)$.

Consider again the proof of Lemma 5.20. In the solution of OPT* to the instance $(\bar{\mu},\sigma)$ each bin contains exactly one item from S^*, which is by construction of the modified instance $(\bar{\mu},\sigma)$ and by definition of the modified BIN COVERING problem. With no subset of $\sigma \setminus S^*$ any bin from $\bar{\mu}$ can be covered since also NFD could not cover the smallest bin from $\bar{\mu}$ with all items from $\sigma \setminus S^*$. Hence, for each item $J_t \in \sigma_R$ that is removed from σ, the revenue of one bin is lost in the solution of OPT*. As each bin from $\bar{\mu}$ has a demand of at least d_{j^*}, there holds $\text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) \leq \text{OPT}^*(\bar{\mu},\sigma) - |\sigma_R|\cdot d_{j^*}$. The latter is equivalent to $\text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) + |\sigma_R|\cdot d_{j^*} \leq \text{OPT}^*(\bar{\mu},\sigma)$. Each bin in μ'' has a demand of at most d_{j^*} and each item from σ_R will reside alone on such a bin. Hence $\text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) + |\sigma_R|\cdot d_{j^*} \geq \text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) + \text{OPT}(\mu'',\sigma_R)$. Finally, $\text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) \geq \text{OPT}^*(\mu^*,\sigma\setminus\sigma_R) \geq \text{OPT}(\mu^*,\sigma\setminus\sigma_R) \geq \text{OPT}(\mu^*,\sigma_2')$ because $\sigma\setminus\sigma_R \supseteq \sigma_2'$. We remark that $\text{OPT}^*(\bar{\mu},\sigma\setminus\sigma_R) \geq \text{OPT}^*(\mu^*,\sigma\setminus\sigma_R)$ holds because we have shown in Lemma 5.20 that $\text{OPT}^*(\bar{\mu},\sigma) \geq \text{OPT}^*(\mu^*,\sigma)$, and by construction of $\bar{\mu}$ and definition of the modified BIN COVERING problem this inequality also holds for any subset of σ. This shows the claim. □

Lemma 5.25. *Let (μ,σ) be an instance such that* NFD *gives a solution with $k \geq 2$ well-covered bins. If at least one of these bins contains at least three items, then* $\text{OPT}(\mu,\sigma)/\text{NFD}(\mu,\sigma) \leq 9/4$.

Proof. Note that the case $k \geq 4$ is already covered by Observation 5.19 and hence we only have to argue about the cases $k = 2$ and $k = 3$. Let $M_{j_1},\ldots,M_{j_k}$ be the bins that are well-covered in the solution of NFD, and assume that these bins are covered with the items $J_1,\ldots,J_{t^*}$. Recall that $\ell(S_j) \leq 2d_j$ for all $j \in \{j_1,\ldots,j_k\}$, by definition of well-coverage. Assume bin M_{j^*} is well-covered and contains $t' \geq 3$ items, where the existence of M_{j^*} is guaranteed by precondition. Then even $\ell(S_{j^*}) \leq t'/(t'-1)d_{j^*}$, by Observation 5.12. Because $t' \geq 3$, there holds $\ell(S_{j^*}) \leq 3/2 d_{j^*}$.

By definition of well-coverage it follows $\ell(S_{j_k+1}) = 0$. Hence $\sum_{t=t^*+1}^{n} p_t < d_{j_k+1} \leq d_{j_k}$ because NFD would have covered bin M_{j_k+1} otherwise.

We bound

$$\begin{aligned}
\sum_{t=1}^{n} p_t &= \sum_{t=1}^{t^*} p_t + \sum_{t=t^*+1}^{n} p_t \\
&= \sum_{\substack{t:J_t \in S_j, \\ j \in \{j_1,\ldots,j_k\}}} p_t + \sum_{t=t^*+1}^{n} p_t \\
&= \sum_{\substack{j \in \{j_1,\ldots,j_k\}, \\ j \neq j^*}} \ell(S_j) + \ell(S_{j^*}) + \sum_{t=t^*+1}^{n} p_t \\
&< \sum_{\substack{j \in \{j_1,\ldots,j_k\}, \\ j \neq j^*}} \ell(S_j) + \ell(S_{j^*}) + d_{j_k} \\
&\leq \sum_{\substack{j \in \{j_1,\ldots,j_k\}, \\ j \neq j^*}} 2d_j + 3/2d_{j^*} + d_{j_k} \\
&\leq \sum_{j \in \{j_1,\ldots,j_{k-1}\}} 2d_j + 5/2d_{j_k}.
\end{aligned}$$

As $\text{OPT}(\mu,\sigma) \leq \sum_{t=1}^{n} p_t$, there holds $\text{OPT}(\mu,\sigma) \leq 2d_{j_1} + \cdots + 2d_{j_{k-1}} + 5/2d_{j_k}$ by the above. Further, $\text{NFD}(\mu,\sigma) \geq d_{j_1} + \cdots + d_{j_k}$.

Firstly consider the case $k = 3$. Then $\text{OPT}(\mu,\sigma) \leq 2d_{j_1} + 2d_{j_2} + 5/2d_{j_3}$ and $\text{NFD}(\mu,\sigma) \geq d_{j_1} + d_{j_2} + d_{j_3}$. Hence $\text{OPT}(\mu,\sigma)/\text{NFD}(\mu,\sigma) = 2 + \frac{1/2d_{j_3}}{d_{j_1}+d_{j_2}+d_{j_3}} \leq 2 + 1/2d_{j_3}/(3d_{j_3}) = 2 + 1/6 \leq 9/4$.

If $k = 2$, then $\text{OPT}(\mu,\sigma) \leq 2d_{j_1} + 5/2d_{j_2}$ and $\text{NFD}(\mu,\sigma) \geq d_{j_1} + d_{j_2}$. Hence $\text{OPT}(\mu,\sigma)/\text{NFD}(\mu,\sigma) = 2 + \frac{1/2d_{j_2}}{d_{j_1}+d_{j_2}} \leq 2 + 1/2d_{j_2}/(2d_{j_2}) = 9/4$. This concludes the proof of the lemma. □

We emphasize that Lemma 5.25 does not require that the solution of NFD contains no non-empty bins that are not well-covered.

Lemma 5.26 (Decomposition Lemma)**.** *Let (μ,σ) be an instance such that* NFD *gives a solution S containing k well-covered bins and at least one filled bin that is not well-covered. Let M_{j^*} be the head of the instance. Let $\sigma_1' = \bigcup_{1 \leq j \leq j^*} S_j$ and $\sigma_1'' = \sigma \setminus \sigma_1'$. Further let $\mu' = \{M_1,\ldots,M_{j^*}\}$ and $\mu'' = \mu \setminus \mu'$. Then* $\text{OPT}(\mu,\sigma)/\text{NFD}(\mu,\sigma) \leq \max\{9/4, \text{OPT}(\mu'',\sigma_1'')/\text{NFD}(\mu'',\sigma_1'')\}$.

Proof. At first we may assume that the number of well-covered bins, k, is at most three since otherwise the claim already follows by Observation 5.19.

Consider the case that all of the well-covered bins in S contain at most two items. Observe that $\text{NFD}(\mu, \sigma_1) = \text{NFD}(\mu', \sigma_1') + \text{NFD}(\mu'', \sigma_1'')$ holds by the definition of these sets. Consider an optimal solution O. Let $\sigma_2' = \bigcup_{1 \leq i \leq i^*} O_i$ be the set of items that reside on the bins in μ' and $\sigma_2'' = \sigma \setminus \sigma_2'$. Clearly, $\text{OPT}(\mu, \sigma_2) = \text{OPT}(\mu, \sigma_2') + \text{OPT}(\mu'', \sigma_2'')$.

We show that $\text{OPT}(\mu'', \sigma_2'') \leq \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'') + \text{OPT}(\mu'', \sigma_1'')$. We want to emphasize this is not a trivial relation since it could be that OPT has to use all items from σ_2'' to cover one bin from μ'' and σ_1'' contains only one item from σ_2'' that is not large enough to cover a bin. Then we had $\text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'') + \text{OPT}(\mu'', \sigma_1'') = 0$ and $\text{OPT}(\mu'', \sigma_2'') > 0$. Nevertheless, here we are in the situation that $\sigma_2'' \setminus \sigma_1'' \subseteq \sigma_1'$. Thus all items in $\sigma_2'' \setminus \sigma_1''$ are at least as big as the smallest item in σ_1', which is the item on bin M_{j^*} in S. Hence for every item $J_t \in (\sigma_2'' \setminus \sigma_1'') \subseteq \sigma_1'$ we have that $p_t \geq 2d_j$ for every bin $M_j \in \mu''$. Thus every such an item resides alone on a bin in an optimal solution to the instance (μ'', σ_2'') (and also fills the bin on which it resides), and this proves that the inequality holds.

With this

$$\begin{aligned}
\frac{\text{OPT}(\mu, \sigma)}{\text{NFD}(\mu, \sigma)} &= \frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_2'')}{\text{NFD}(\mu', \sigma_1') + \text{NFD}(\mu'', \sigma_1'')} \\
&\leq \frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'') + \text{OPT}(\mu'', \sigma_1'')}{\text{NFD}(\mu', \sigma_1') + \text{NFD}(\mu'', \sigma_1'')} \\
&\leq \max\left\{ \frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'')}{\text{NFD}(\mu', \sigma_1')}, \frac{\text{OPT}(\mu'', \sigma_1'')}{\text{NFD}(\mu'', \sigma_1'')} \right\} \qquad (5.19) \\
&\leq \max\left\{ \frac{9}{4}, \frac{\text{OPT}(\mu'', \sigma_1'')}{\text{NFD}(\mu'', \sigma_1'')} \right\}, \qquad (5.20)
\end{aligned}$$

where we have used elementary calculus in (5.19) and Lemma 5.24 in (5.20).

Now consider the case that one of the well-covered bins contains at least three items. For $k \geq 2$ the claim follows by Lemma 5.25 and we are left to show the claim for $k = 1$.

If an optimal algorithm decides to assign one of the items residing on the only well-covered bin M_1 to a bin M_j, with $j \geq j^*$, then the claim follows by Lemma 5.23. Hence it suffices to argue about the case $\text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'') = \text{OPT}(\mu'', \sigma_2'' \setminus (S_1 \cup \sigma_1''))$.

Again we decompose the solutions of NFD and OPT identically as above in the case in which each well-covered bin contains at most two items, and we are left to show Step (5.20). Let j' be the smallest index of a bin that contains items

and is not well-covered in the solution of NFD. Define $\mu^* = \{M_1, \ldots, M_{j'-1}\}$. Then $\text{NFD}(\mu', \sigma_1') = \text{NFD}(\mu^*, \sigma_1') + \sum_{j' \le j \le j^*} d_j$ and $\text{OPT}(\mu', \sigma_2') \le \text{OPT}(\mu^*, \sigma_2') + \sum_{j' \le j \le j^*} d_j$. We find

$$\begin{aligned}
&\frac{\text{OPT}(\mu', \sigma_2') + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'')}{\text{NFD}(\mu', \sigma_1')} \\
&\le \frac{\text{OPT}(\mu^*, \sigma_2') + \sum_{j' \le j \le j^*} d_j + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'')}{\text{NFD}(\mu^*, \sigma_1') + \sum_{j' \le j \le j^*} d_j} \\
&\le \max\left\{ \frac{\text{OPT}(\mu^*, \sigma_2')}{\text{NFD}(\mu^*, \sigma_1')}, \frac{\sum_{j' \le j \le j^*} d_j + \text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'')}{\sum_{j' \le j \le j^*} d_j} \right\},
\end{aligned}$$

where we have used elementary calculus in the last step. By the case

$$\begin{aligned}
\frac{\text{OPT}(\mu'', \sigma_2'' \setminus \sigma_1'') + \sum_{j' \le j \le j^*} d_j}{\sum_{j' \le j \le j^*} d_j} &= \frac{\text{OPT}(\mu'', \sigma_2'' \setminus (S_1 \cup \sigma_1'')) + \sum_{j' \le j \le j^*} d_j}{\sum_{j' \le j \le j^*} d_j} \\
&\le \frac{2 \cdot \sum_{j' \le j \le j^*} d_j}{\sum_{j' \le j \le j^*} d_j} = 2,
\end{aligned}$$

where in the second to last step we have used Property (5.12) in Lemma 5.25. Furthermore,

$$\frac{\text{OPT}(\mu^*, \sigma_2')}{\text{NFD}(\mu^*, \sigma_1')} = \frac{\text{OPT}(\mu^*, \sigma_2')}{\text{NFD}(\mu^*, \sigma_1' \cup \sigma_2')} \le \frac{\text{OPT}(\mu^*, \sigma_1' \cup \sigma_2')}{\text{NFD}(\mu^*, \sigma_1' \cup \sigma_2')} \le 9/4,$$

using Lemma 5.21 in the last step. □

Note that the above decomposition lemma gives that we can consider the instance (μ'', σ_1'') in fact as a separate subproblem. Hence this lemma is applicable in a recursive step. We now can prove Theorem 5.10.

Proof. (of Theorem 5.10). First observe that Example 5.9 yields a lower bound of $9/4$ on the approximation ratio of NFD. Let k be the number of well-covered bins in the instance. If $k \ge 4$, then Observation 5.19 already gives the claim. Thus let $k \in \{1, 2, 3\}$. Assume firstly that there is no additional filled bin besides the k well-covered bins. If one of the k bins contains at least three items, then the claim follows from Lemma 5.21 and Lemma 5.25. If every one of the k well-covered bins contains at most two items the statement follows from Lemma 5.20.

Now let there be $k \in \{1, 2, 3\}$ well-covered bins in the instance and at least one filled but not well-covered bin in the instance. Define $\mu' = \{M_1, \ldots, M_{j^*}\}$,

$\mu'' = \mu \setminus \mu'$, where M_{j^*} is the head of the instance, and define σ' to be the set of items that are assigned to the bins in μ' by NFD and $\sigma'' = \sigma \setminus \sigma'$. Now we can apply Lemma 5.26. Observe (μ'', σ'') is a smaller instance, with at least one not well-covered bin less. Hence we can apply the analysis in a recursive step on the instance (μ'', σ''). The recursion terminates if (μ'', σ'') is an instance in which there are only well-covered bins or in which the solution of NFD has no covered bins. Clearly, in the latter case we have that NFD is optimal and in the former we can argue as above. The algorithm can be implemented such that the running-time is dominated by sorting bins and items. □

Monotonicity of Next Fit Decreasing for Variable-Sized Bin Covering

The following property of NFD is obtained as a side-result using the techniques employed in the analysis of the approximation ratio.

Property 5.27. NFD *is a monotone algorithm for* VARIABLE-SIZED BIN COVERING*, i. e. if* (μ, σ') *is an instance and* $\sigma' \subseteq \sigma$*, then it follows* $\text{NFD}(\mu, \sigma) \geq \text{NFD}(\mu, \sigma')$.

Proof. Obviously it suffices to show the claim when the instance (μ, σ) contains exactly one new item in comparison to the instance (μ, σ'), i. e. there is some $J_{t^*} \in \sigma$ with $J_{t^*} \notin \sigma'$ and $\sigma = \sigma' \cup \{J_{t^*}\}$.

If all bins M_j that are filled in the instance (μ, σ') are filled as well in the instance (μ, σ), then the claim follows. Thus assume there is a bin $M_{j'}$ with smallest index that is filled in the instance (μ, σ') but is not in the instance (μ, σ). Let M_{j^*} be a bin with smallest index that is covered in the instance (μ, σ) but is not in the instance (μ, σ'). Such a bin exists and there holds $j^* < j'$ for the following reason. Assume to the contrary that each bin M_j, $j < j'$, that is covered in the instance (μ, σ) is also covered in the instance (μ, σ'). Consider bin $M_{j'}$. By choice of $M_{j'}$ each bin M_j, with $j < j'$, is covered in the solution to (μ, σ) if and only if it is covered in the solution to (μ, σ). Recall items are sorted non-increasingly by sizes and assigned in this ordering by NFD. Hence when considering bin $M_{j'}$, it follows that NFD on the instance (μ, σ) has a superset of the items left that NFD on the instance (μ, σ') has left. Thus NFD working on the item set σ can cover bin $M_{j'}$ as well, which contradicts the existence of $M_{j'}$. Thus the bin M_{j^*} with $j^* < j'$ exists.

It follows for every bin $M_1, \ldots, M_{j^*-1}$ that each one of these is covered in both instances or is covered in none of these. Let J_{t^*} be the item with smallest index that resides on a bin M_j, with $j \geq j^*$, in the instance (μ, σ'). As $M_{j'}$ is covered, item J_{t^*} exists. Since NFD did not cover bin M_{j^*} in the instance (μ, σ'),

we have $d_{j^*} > \sum_{t=t^*}^{n} p_t$. This already gives the claim for the following reasons. The bins $M_1, \dots, M_{j^*-1}$ were identically covered in both instances. In instance (μ, σ) additionally at least bin M_j^* is covered, which yields revenue d_{j^*}, and the revenue that can be gained by covering some of the bins $M_{j^*+1}, \dots, M_m$ in the instance (μ, σ') is smaller than d_{j^*}, as just shown. □

5.3.2 Inapproximability in the Unit Supply Model

By reduction from PARTITION it is not hard to see that the classical BIN COVERING is NP-hard and is not approximable within a factor of two, unless P = NP. This clearly extends to all of the models we consider here. Now the question arises if improvements in an asymptotic notion, where the optimal revenue diverges, are possible. Note that we still require $r_j = d_j$, which yields that divergence of the optimal revenue implies divergence of the total demand of the instance. However, it is not obvious how to define a suitable asymptotics in the *unit supply* model: If only the total item size diverges, the optimal revenue does not. If, in addition, the bin demands (but not their number) diverges, these instances still contain PARTITION. Thus we consider an asymptotics, where the total item size, the total demand, and the number of bins in an optimal solution diverges.

Recall that we define $p^+ = \sum_{t=1}^{n} p_t$. The following theorem states that no algorithm can have an approximation ratio of $2 - \varepsilon$ if $\varepsilon > 0$ is any constant, unless P = NP. We remark that we show below Theorem 5.28 that the choice of $p^+ \in \omega(m)$ as needed by this theorem is in fact possible.

Theorem 5.28. *Consider* VARIABLE-SIZED BIN COVERING *with unit supply. For every $2 \leq m \leq n$ there is an instance (μ, σ), with $|\mu| = m$ and $|\sigma| = n + m - 2$, such than in an optimal solution m bins are covered but there is no polynomial time algorithm with approximation factor better than $\rho = 2 - \frac{m-2}{p^+/2}$, unless P = NP.*

Proof. We use a reduction from the PARTITION problem. Recall, in PARTITION we are given a set of items $\hat{P} = \{\hat{P}_1, \dots, \hat{P}_n\}$, where item $\hat{P}_t$ has integral size $\hat{p}_t$. Again, let $\hat{p}^+ = \sum_{t=1}^{n} {\hat{p}_t}^+$ as a shorthand. Our goal is to answer whether there exists a subset $P' \subset \hat{P}$ such that $\sum_{t:\hat{P}_t \in P'} \hat{p}_t = \hat{p}^+/2$, i. e. if the items from $\hat{P}$ can be partitioned into two sets, P' and $\hat{P} \setminus P'$, of equal total size.

Let $\hat{P}$ be a PARTITION instance. We define an instance (μ, σ) for VARIABLE-SIZED BIN COVERING. We set $\mu = \{M_1, \dots, M_m\}$ and $\sigma = \{J_1, \dots, J_{m+n-2}\}$ with $p_t = 2\hat{p}_t m$, for $1 \leq t \leq n$, and $p_t = 1$ for $n + 1 \leq t \leq n + m - 2$, i. e. the items of the PARTITION instance are scaled by a factor of $2m$, which can clearly be done in polynomial time. Set $d_1 = m\hat{p}^+$, $d_2 = m\hat{p}^+$, where we may assume $\hat{p}^+$ is integral. Further, we set $d_3 = \dots = d_m = 1$ and $r_j = d_j$ for all $1 \leq j \leq m$.

Now we see that the solution of VARIABLE-SIZED BIN COVERING has a value of $2m\hat{p}^+ + m - 2$ if the PARTITION problem has a solution. If the PARTITION problem has no solution, then the value of the solution to VARIABLE-SIZED BIN COVERING is at most $m\hat{p}^+ + m - 2$, as we argue now. Consider firstly the case that all items $J_1, \ldots, J_n$ are assigned to bins M_1 and M_2 by an algorithm and the items $J_{n+1}, \ldots, J_{n+m-2}$ are assigned to bins $M_3, \ldots, M_m$. In the PARTITION problem, we have for every choice of $P' \subset \hat{P}$ that

$$\sum_{t:\hat{P}_t \in P'} \hat{p}_t \neq \sum_{t:\hat{P}_t \in \hat{P} \setminus P'} \hat{p}_t,$$

i. e. the left-hand sum and the right-hand sum differ by at least one, which is because the $\hat{p}_t$ are integral. Hence in the instance for VARIABLE-SIZED BIN COVERING, which uses the scaled sizes, we have $\ell(1)$ and $\ell(2)$ differ for every assignment of the items $J_1, \ldots, J_n$ to bins M_1 and M_2 by at least $2m$. Let w. l. o. g. be $\ell(1) > \ell(2)$, then we have $\ell(1) - \ell(2) \geq 2m$, and thus $\ell(2) \leq 2m\hat{p}^+/2 - m$. Consequently, even if all items $J_{n+1}, \ldots, J_{n+m-2}$ are put by an algorithm on bin M_2, we have

$$\ell(2) \leq 2m\hat{p}^+/2 - m + (m-2) = 2m\hat{p}^+/2 - 2 < m\hat{p}^+,$$

and we see, bin M_2 is not covered if the items $J_{n+1}, \ldots, J_{n+m-2}$ are assigned arbitrarily. Hence, in this case, the gained revenue is bounded by $d_1 + d_3 + d_4 + \cdots + d_m \leq m\hat{p}^+ + m - 2$. If, in addition, the items $J_1, \ldots, J_n$ are assigned to arbitrary bins, the value of the solution may only decrease, and we have shown that the value of a solution on the given instance is at most $m\hat{p}^+ + m - 2$ if the PARTITION problem has no solution.

Hence, an algorithm with approximation ratio smaller than $\rho = 2 - \frac{m-2}{m\hat{p}^+ + m - 2} \geq 2 - \frac{m-2}{p^+/2}$ can distinguish the cases and solve the PARTITION problem. □

Note that $m \leq n$, where n is the number of items in the PARTITION instance $\hat{P}$ and m the number of bins in the above construction. Hence the size of the PARTITION instance is still polynomially bounded in n if we choose $\hat{p}^+ \in \Theta(2^m)$. Thus, the choice $p^+ = \omega(m)$ is indeed possible, which proves that $\rho \to 2$ when $m \to \infty$.

We remark that the hardness stated in Theorem 5.28 also transfers to the infinite supply model if one considers the analogue asymptotic notion, where the total demand of (some) bin *types* diverges. This is the difference between that notion and the notion captured by $\bar{\rho}$, where bin type demands are assumed to be fixed.

5.4 Variable-Sized Bin Covering in the Infinite Supply Model

In this section we consider the infinite supply model for VARIABLE-SIZED BIN COVERING. Recall that in this model we have arbitrarily many bins of each type M_j available. The result of this section is an AFPTAS.

It turns out that the APTAS of Csirik et al. [23] and the method of Jansen and Solis-Oba [42] for the classical BIN COVERING with infinite supply can be extended. The basic idea for adapting the PTAS of [23] is to ignore bin types with small demand. This idea was also used in BIN PACKING [48]. Adjusting the parameters in the algorithm of [23] and adapting the calculations gives the desired result. Then we can also extend the algorithm of [42]. Adapting the proofs, this yields the desired result.

It turns out that normalizing the demands of bins (and the sizes of items) is advantageous here. Thus we assume throughout this section $1 = d_1 > \cdots > d_m > 0$. Since we are in the VARIABLE-SIZED BIN COVERING model, we have $d_j = r_j$ for all $j = 1, \ldots, m$.

5.4.1 An Asymptotic Polynomial-Time Approximation Scheme

In this section we present an asymptotic polynomial-time approximation scheme.

Outline of the APTAS. Let $\varepsilon > 0$ be the desired approximation factor and we assume w. l. o. g. that $1/\varepsilon$ is integral. In the algorithm we delete all bin types with demand at most ε. Then we partition the items σ into three sets: σ_L the set of large items, σ_M the set of medium-sized items, and σ_T the set of tiny items (for a formal definition of σ_L, σ_M, and σ_T, see Algorithm 5.3. The large items σ_T are further subdivided into $1/\varepsilon^4$ groups, where each group has (almost) equal size w. r. t. the number of items it contains (cf. Step 3 of Algorithm 5.3). This grouping technique originates from a paper of Fernandez de la Vega and Lueker [30].

In each group all items are rounded down to the size of the smallest item of the respective group. Note, this implies that there are at most $k = 1/\varepsilon^4$ many different sizes for the large items. The idea is here that the items of group i can replace the items from group $i + 1$ in a solution. By this procedure only a revenue bounded by the size of the first group is lost. Then all possible assignments – referred to as configurations in the following – of the rounded down large items to bins are enumerated.

Via an appropriate linear program formulation a solution is determined. More precisely, a solution to a linear program gives how many bins of each type are assigned items and according to which configuration they are assigned items. Here it is crucial that the linear program formulation ensures that only such sets of

configurations in the solution are used so that the non-large items (i. e. the items from $\sigma_M \cup \sigma_T$) can fill the possible only partially covered bins in a greedy way. A description of the algorithm can be found in Figure 5.3; the notations used therein are defined below. We obtain the following result.

Theorem 5.29. *There is an APTAS for the* VARIABLE-SIZED BIN COVERING *problem in the infinite supply model.*

Introductory Definitions. The following notation is used in the description of the APTAS in Figure 5.3.

- Call a vector $v \in \{0, \ldots, n\}^k$ a configuration, where $k = 1/\varepsilon^4$ is an integral constant. Let $s = (s_1, \ldots, s_k)$ be the vector of large sizes, i. e. the size of the items in the respective group. The exact values s_i are determined in Step 3 of the Algorithm 5.3.

- Let $e_i \in \{0, 1\}^k$ the (row) vector with an entry 1 at position i and 0 at all positions $i' \neq i$.

- Let $n(v, i) = e_i \cdot v$ be the number of items of size s_i in configuration v.

- Observe that $s \cdot v$ is the total size of all items that are in configuration v. Let $C_j = \{v \in \{0, \ldots, n\}^k \mid d_{j-1} > s \cdot v \geq d_j\}$, where $d_0 := \infty$, i. e. associate every configuration v to a bin type M_j of largest demand such that configuration v covers a bin of type M_j and refer with C_j to the set of configurations associated to bin type M_j.

- Let $r(v, j) = d_j - s \cdot v$ be the demand of a bin of type M_j that is not covered by configuration v (the remainder).

- Let $\tilde{C}_j = \{v \in \{0, \ldots, n\}^k \mid r(v, j) > 0\}$, i. e. $\tilde{C}_j$ contains all configurations that do not cover a bin of type M_j.

- Let $n(i)$ be the number of items of size s_i, for $1 \leq i \leq k$, in the rounded instance. "Rounded instance" refers here to the instance that is obtained from the input instance by rounding down the item sizes of the large items, as done in Step 3 of Algorithm 5.3.

- For a subset $\sigma' \subseteq \sigma$ let $p^+(\sigma') = \sum_{t: J_t \in \sigma'} p_t$ denote the total size of the items in σ', where we use as before $p^+ := p^+(\sigma)$ as a shorthand.

Let $\text{OPT}(\mu; \sigma_L, \sigma_T)$ denote the value of an optimal solution in which the items σ_L and σ_T are assigned to bins from the types contained in μ, where the items σ_T may be split arbitrarily. Linear Program LP (5.21) describes this relaxation on the

instance $(\mu; \sigma_L, \sigma_T)$, and is used as a subprocedure by the APTAS. The variables y_v and $z_{v',j'}$ correspond to configurations such that $v \in C_j$ and $v' \in \tilde{C}_{j'}$. This LP is solved as a subprocedure by the APTAS.

1. Remove all bin types with demand at most ε and refer with (μ', σ) to the modified instance from now on and let $|\mu'| = m$.
2. Sort items non-increasingly. If $n < \lceil p^+/\varepsilon^3 \rceil + \lfloor p^+/\varepsilon \rfloor$, then set $\sigma_L = \{J_1, \ldots, J_n\}$ to be the set of large items and $\sigma_M = \sigma_T = \emptyset$ to be the sets of medium and tiny items. Else define the sets σ_L, σ_M, σ_T of large, medium and tiny items as $\sigma_L = \{J_1, \ldots, J_{\lceil p^+/\varepsilon^3 \rceil}\}$, $\sigma_M = \{J_{\lceil p^+/\varepsilon^3 \rceil + 1}, \ldots, J_{\lceil p^+/\varepsilon^3 \rceil + \lfloor p^+/\varepsilon \rfloor}\}$ and $\sigma_T = \{J_{\lceil p^+/\varepsilon^3 \rceil + \lfloor p^+/\varepsilon \rfloor + 1}, \ldots, J_n\}$.
3. Subdivide the items of σ_L into $k = 1/\varepsilon^4$ groups as follows. Let $\alpha = |\sigma_L| \text{ div } k$ and $\beta = |\sigma_L| \mod k$. Groups $1, \ldots, \beta$ contain $\alpha + 1$ items each and the groups $\beta + 1, \ldots, k$ contain α items each. In every group i, we round down the size of every item J_t of that group to the size $p_t := s_i$, where s_i is determined to be the size of the smallest item in group i.
4. Enumerate all configurations and compute C_j and $\tilde{C}_j$ for $j = 1, \ldots, m$.
5. Introduce variables y_v such that for each $v \in C_j$ the variable y_v is associated to configuration v. Introduce variables $z_{v,j}$ for each $v \in \tilde{C}_j$ such that each $z_{v,j}$ is associated to configuration v and bin type M_j.
6. Compute $p^+(\sigma_T)$ and $n(i)$ for $i = 1 \ldots, k$. Solve LP (5.21).
7. Set for every variable y_v and $z_{v,j}$ of LP (5.21) $y'_v := \lfloor y_v \rfloor$ and $z'_{v,j} = \lfloor z_{v,j} \rfloor$.
8. Construct a solution in the following way.
   (a) For every configuration $v \in \{0, \ldots, n\}^k$ assign items to y'_v many bins of type M_j according to the configuration v, where M_j is the unique bin type associated to v; recall M_j is associated to v if $v \in C_j$.
   (b) For every pair (v, j) assign items according to configuration v to $z'_{v,j}$ many bins of type M_j.
   (c) Cover the bins created according to the $z'_{v,j}$ variables in a greedy way – for example with NFD – using the items from $\sigma_M \cup \sigma_T$, where we are left to show that this is possible.

Figure 5.3: Description of the APTAS.

$$\text{maximize} \qquad \sum_{j=1}^{m} d_j \Big(\sum_{v \in C_j} y_v + \sum_{v \in \widetilde{C}_j} z_{v,j}\Big) \tag{5.21}$$

$$\text{subject to} \sum_{v \in \{1,\ldots,n\}^k} n(v,i) \left(y_v + \sum_{j=1}^{m} z_{v,j} \right) \leq n(i) \qquad 1 \leq i \leq k$$

$$\sum_{j=1}^{m} \sum_{v \in \widetilde{C}_j} r(v,j) z_{v,j} \leq p^+(\sigma_T)$$

$$y_v \geq 0 \qquad v \in \{0,\ldots,n\}^k$$

$$z_{v,j} \geq 0 \qquad v \in \{0,\ldots,n\}^k, \quad 1 \leq j \leq m$$

Now, we give the key observation that lets us adapt the algorithm from [23] to the VARIABLE-SIZED BIN COVERING model with infinite supply of bins.

Observation 5.30. *Fix an instance (μ, σ). Let $\mu' := \{M_j \in \mu \mid d_j > \varepsilon\}$ be the set of bin types that have demand more than ε. Then $(1+\varepsilon)\text{OPT}(\mu', \sigma) + 1 \geq \text{OPT}(\mu, \sigma)$.*

Proof. Consider an optimal solution O to the instance (μ, σ), and we may assume that only bins with demand at most ε are covered; otherwise the bound on $\text{OPT}(\mu', \sigma)$ can only be better. We partition the items residing on bins of types in $\mu \setminus \mu'$ into two sets σ_1 and σ_2. Set σ_1 contains the items having size smaller than ε, and set σ_2 contains the items of size at least ε. Observe that in an optimal solution to the instance (μ, σ) the items from σ_1 and σ_2 will reside on distinct bins. Hence it suffices to show for $i = 1$ and $i = 2$ that $(1+\varepsilon)\text{OPT}(\mu', \sigma_i) \geq \text{OPT}(\mu, \sigma_i)$, and we may further lose one additional bin in total.

Clearly, if we put all items from σ_1 with NFD on bins of demand 1, then for every filled bin M_j we have $\ell(j) < 1 + \varepsilon$. Hence, if $p^+(\sigma_1)$ is the overall size of all items in σ_1, we yield revenue at least $\lfloor p^+(\sigma_1)/(1+\varepsilon) \rfloor \geq p^+(\sigma_1)/(1+\varepsilon) - 1$ whilst $p^+(\sigma_1)$ is an upper bound for the revenue that in an optimal solution to the partial instance (μ, σ_1) can be gained.

We also assign the items from σ_2 with NFD to bins of demand 1. If there are unassigned items, then we assign them together with the unassigned items from σ_1 to a bin of demand 1. Thus after both steps, items of a total size smaller than 1 remain unassigned, and it suffices to show that we gain a revenue of at least a $1/(1+\varepsilon)$ fraction of the revenue that is gained on the instance (μ, σ_2) in an optimal solution.

Consider a bin M_j that is covered by t^* many items from σ_2 in the solution S of NFD to the instance (μ', σ_2). If M_j has fill level $\ell(S_j) \leq 1 + \varepsilon$, the claim follows. Thus assume $\ell(S_j) > 1 + \varepsilon$ and let $\varepsilon' = \ell(S_j) - 1$. Let μ_j be the set of bins to which OPT assigned the items from S_j and recall that $|\mu_j| = |S_j| = t^*$, i. e. every item resides alone on its bin in an optimal solution. Since we have $\varepsilon' > \varepsilon$ and the last item assigned to M_j was a smallest on this bin, we conclude that all items $J_t \in S_j$ have a size of at least ε'. Thus for all $M_{j'} \in \mu_j$ there holds that $\ell(O_{j'}) - d_{j'} > \varepsilon' - \varepsilon$, that is also OPT "wastes" more than a size of $\varepsilon' - \varepsilon$ per item that resides on a bin $M_{j'} \in \mu_j$. Since $|\mu_j| = t^*$, it follows that the revenue gained in an optimal solution for every such bin M_j that NFD covers with $\ell(S_j) = 1 + \varepsilon' > 1 + \varepsilon$, is bounded by

$$1 + \varepsilon' - t^*(\varepsilon' - \varepsilon) = 1 - \varepsilon'(t^* - 1) + \varepsilon t^* \leq 1 - \varepsilon(t^* - 1) + \varepsilon t^* = 1 + \varepsilon.$$

The revenue NFD yields for these items is in total at least 1, and hence the claim follows. □

Observation 5.31. *There holds* $p^+ \leq 2\text{OPT}(\mu, \sigma_L \cup \sigma_M \cup \sigma_T) + 2$.

Proof. We can assume w. l. o. g. that the largest items in the instance have size less than 1 since otherwise a preprocessing can remove larger items and assign them to the bin type with demand 1, which is clearly optimal. Then it is easy to see that $\text{OPT}(\mu, \sigma_L \cup \sigma_T \cup \sigma_M) \geq \lfloor p^+/2 \rfloor$ since already NFD gives such a bound using only the largest bin type with demand 1. Rearranging and taking into account the rounding gives the claim. □

Observation 5.32. *Let* $p^+ \geq 2$ *and* $\varepsilon \leq 1/6$. *If* μ' *is a set of bin types containing only bin types with demand at least* ε, *then* $\text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) \leq \text{OPT}(\mu', \sigma_L \cup \sigma_T)/(1 - 2\varepsilon) + 2$.

Proof. Consider an optimal solution to the instance $(\mu', \sigma_L \cup \sigma_M \cup \sigma_T)$. Since OPT yields at most $\lfloor p^+/\varepsilon \rfloor$ many bins by the smallest demand of a bin type, the average number of large items per bin is at least $1/\varepsilon^2$ by the definition of the set σ_L.

Hence removing $\lfloor p^+\varepsilon \rfloor + 1$ bins with the largest number of large items, removes at least $\lfloor p^+/\varepsilon \rfloor$ many large items. These can now be used instead of the medium-sized items in the rest of the instance since there are at most so many medium items in the instance. The modified solution has at most $\lfloor p^+/\varepsilon \rfloor + 1$ bins less than the optimal solution.

With $\text{OPT}(\mu', \sigma_L \cup \sigma_T \cup \sigma_M) \geq \lfloor p^+/2 \rfloor \geq p^+/2 - 1$ as argued in the proof of Observation 5.31 and as $\varepsilon \leq 1/6$, the removal of the $\lfloor p^+\varepsilon \rfloor + 1$ bins is possible since an optimal solution contains at least this many filled bins. Further we have shown that $\text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) - \lfloor p^+\varepsilon \rfloor - 1 \leq \text{OPT}(\mu', \sigma_L \cup \sigma_T)$.

With this,

$$\begin{aligned}\text{OPT}(\mu', \sigma_L \cup \sigma_T) &\geq \text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) - \lfloor p^+\varepsilon \rfloor - 1 \\ &\geq \text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) - p^+\varepsilon - 1 \\ &\geq \text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) - 2\varepsilon\text{OPT}(\mu', \sigma_L \cup \sigma_T \cup \sigma_M) - 2\varepsilon - 1 \\ &\geq (1-2\varepsilon)\text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) - 2\varepsilon - 1,\end{aligned}$$

where we have used Observation 5.31 in the third line. Rearranging and using $\varepsilon \leq 1/6$, the claim follows. □

Observation 5.33. *If $\varepsilon \leq 1/6$, then $\text{OPT}(\mu, \sigma_L \cup \sigma_M \cup \sigma_T) \leq (1+\varepsilon)/(1-2\varepsilon)\text{OPT}(\mu'; \sigma_L, \sigma_T) + 4$.*

Proof. We have

$$\begin{aligned}&\text{OPT}(\mu, \sigma_L \cup \sigma_M \cup \sigma_T) \\ &\leq (1+\varepsilon)\text{OPT}(\mu', \sigma_L \cup \sigma_M \cup \sigma_T) + 1 && (5.22)\\ &\leq (1+\varepsilon)\left(\frac{1}{1-2\varepsilon}\text{OPT}(\mu', \sigma_L \cup \sigma_T) + 2\right) + 1 && (5.23)\\ &\leq \frac{1+\varepsilon}{1-2\varepsilon}\text{OPT}(\mu'; \sigma_L, \sigma_T) + 4, && (5.24)\end{aligned}$$

where (5.22) is by Observation 5.30, (5.23) is by Observation 5.32, and (5.24) is by the precondition $\varepsilon \leq 1/6$ and because of $\text{OPT}(\mu', \sigma_L \cup \sigma_T) \leq \text{OPT}(\mu'; \sigma_L, \sigma_T)$, which is obvious. □

Observation 5.34. *Let $\varepsilon \leq 1/10$. Consider an optimal solution corresponding to the value $\text{OPT}(\mu'; \sigma_L, \sigma_T)$. If σ'_L denotes the sets of large items that is obtained by rounding down the item sizes from σ_L as done in Step 3 of Algorithm 5.3, then*

$$\text{OPT}(\mu'; \sigma_L, \sigma_T) \leq \frac{1-2\varepsilon}{1-4\varepsilon-2\varepsilon^2}\text{OPT}(\mu'; \sigma'_L, \sigma_T) + 9.$$

Proof. Let σ'_L denote the set of large items after rounding down the corresponding item sizes in Step 3 in Algorithm 5.3. Recall the definition of α in this step.

In any solution, an item from group i can replace an item from group $i+1$ and at most a revenue of $\alpha + 1$ is lost in total. The latter holds true since at most $\alpha + 1$ bins of demand 1 are lost because we cannot use the items from group 1 for covering bins anymore. In fact, if $|\sigma_L|$ is divisible by k, then we lose only the α largest items. Hence we conclude $\text{OPT}(\mu'; \sigma_L, \sigma_T) \leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + \alpha + 1$.

With this, since $\alpha \leq \lceil p^+/\varepsilon^3 \rceil \cdot \varepsilon^4$ by the number of groups, there holds $\text{OPT}(\mu'; \sigma_L, \sigma_T) \leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + p^+\varepsilon + 2$. Hence, we can bound

$$
\begin{aligned}
&\text{OPT}(\mu'; \sigma_L, \sigma_T) \\
&\leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + p^+\varepsilon + 2 \\
&\leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + 2\varepsilon\text{OPT}(\mu, \sigma_L \cup \sigma_M \cup \sigma_T) + 2\varepsilon + 2 && (5.25) \\
&\leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + \frac{1+\varepsilon}{\frac{1}{2\varepsilon} - 1}\text{OPT}(\mu'; \sigma_L, \sigma_T) + 10\varepsilon + 2 && (5.26) \\
&\leq \text{OPT}(\mu'; \sigma'_L, \sigma_T) + \frac{1+\varepsilon}{\frac{1}{2\varepsilon} - 1}\text{OPT}(\mu'; \sigma_L, \sigma_T) + 3, && (5.27)
\end{aligned}
$$

where we used Observation 5.31 in (5.25), Observation 5.33 in (5.26) and the fact that $\varepsilon \leq 1/10$ in Step (5.27). Rearranging terms and using again $\varepsilon \leq 1/10$ gives the claim. □

Proof. (*of Theorem 5.29*). Assume $\varepsilon \leq 1/10$. We firstly show that all bins output by the algorithm are filled, then we bound the approximation ratio.

As argued in Observation 5.34, we can obtain a solution for the original problem from the solution with the rounded down large items. Hence it suffices to show that all bins that were assigned items in Step 7 are covered.

The bins that were assigned items according to configurations associated to the variables y'_v are filled by definition of the y_v variables. Bins corresponding to the $z'_{v,j}$ variables are filled in Step 8 (c) as we argue now. By the demand of the smallest bins any solution contains at most $\lfloor p^+/\varepsilon \rfloor$ many bins. Consider a bin M_j that is assigned items with a total size of $\ell(j) \geq d_j$ after Step 8 (c). We say that a revenue of $\ell(j) - d_j$ is lost. Since NFD does not assign items to covered bins, it follows that a revenue of at most $\lfloor p^+/\varepsilon \rfloor$ times the size of the largest items used to cover the bins in Step 8 (c) is lost. By definition of set σ_M it follows that most a revenue of $p^+(\sigma_M)$ is lost. Since $p^+(\sigma_T)$ is at most the demand needed to cover the bins corresponding to the $z'_{v,j}$ variables as enforced by the constraints of LP (5.21), and $p^+(\sigma_M \cup \sigma_T)$ is the sum of the items sizes that is available to NFD in order to fill the not covered bins induced by the $z'_{v,j}$ variables, we find that NFD can fill all those bins.

We now give the calculation for the approximation factor, and explain the steps

thereafter. Let $\mu' = \{M_j \in \mu \mid d_j > \varepsilon\}$. We have

$$\begin{aligned}
&\text{OPT}(\mu, \sigma_L \cup \sigma_M \cup \sigma_T) \\
&\leq \frac{1+\varepsilon}{1-2\varepsilon}\text{OPT}(\mu'; \sigma_L, \sigma_T) + 4 && (5.28)\\
&\leq \frac{1+\varepsilon}{1-2\varepsilon}\left(\frac{1-2\varepsilon}{1-4\varepsilon-2\varepsilon^2}\text{OPT}(\mu'; \sigma'_L, \sigma_T) + 9\right) + 4 && (5.29)\\
&\leq \frac{1+\varepsilon}{1-4\varepsilon-2\varepsilon^2}\text{OPT}(\mu'; \sigma'_L, \sigma_T) + 17 && (5.30)\\
&\leq \frac{1+\varepsilon}{1-4\varepsilon-2\varepsilon^2}\left(\sum_{j=1}^{m} d_j\Big(\sum_{v\in C_j} y_v + \sum_{v\in \widetilde{C}_j} z_{v,j}\Big)\right) + 17 && (5.31)\\
&\leq \frac{1+\varepsilon}{1-4\varepsilon-2\varepsilon^2}\left(\sum_{j=1}^{m} d_j\Big(\sum_{v\in C_j} y'_v + \sum_{v\in \widetilde{C}_j} z'_{v,j}\Big) + 1/\varepsilon^4\right) + 19 && (5.32)
\end{aligned}$$

In (5.28) we use Observation 5.33. In (5.29) we apply Observation 5.34. Inequality (5.30) uses the fact that $\varepsilon \leq 1/10$. Inequality (5.31) is easy to observe and finally, in (5.32) we take into account the loss for rounding down the variables of LP (5.21), which is explained as follows.

We have that LP (5.21) has only $1+1/\varepsilon^4$ constraints besides the non-negativity constraints. Thus an optimal basic solution has at most $1 + 1/\varepsilon^4$ fractional values and hence we lose at most so many bins with demand 1 due to rounding down the fractional variables.

Such a solution can be found in polynomial time since LP (5.21) has polynomial size in m and n (though exponentially in $1/\varepsilon$, which is a constant), the bound follows. If $1 + \varepsilon' > 1$ is the desired approximation ratio we set $\varepsilon = \varepsilon'/13$ and run our algorithm which gives an approximation ratio of at least $1 + \varepsilon'$ minus a constant term. Also observe, by our choice of ε, and $\varepsilon' \leq 1$ w. l. o. g., the assumption $\varepsilon \leq 1/10$ was justified. □

5.4.2 An Asymptotic Fully Polynomial-Time Approximation Scheme

Jansen and Solis-Oba [42] gave an AFPTAS for the classical BIN COVERING problem. In this section we extend their method to work for VARIABLE-SIZED BIN COVERING in the infinite supply model, which will prove Theorem 5.35.

Theorem 5.35. *There is an AFPTAS for* VARIABLE-SIZED BIN COVERING *in the infinite supply model.*

Formulation as Resource Sharing Problem and Overall Method

The AFPTAS does not solve LP (5.21) in Step 6 of the APTAS 5.3. Instead this LP is approximated. We will show later how to transform such an approximate solution into a feasible solution for LP (5.21). Then we apply the rounding procedure from Theorem 5.29. Recall that $k = 1/\varepsilon^4$ is the number of different large sizes and $(n+1)^k$ is the number of configurations. For ease of notation let $|C| := (n+1)^k$. Let $x = (y_1, \ldots, y_{|C|}, z_{1,1}, \ldots, z_{1,m}, z_{2,1}, \ldots, z_{2,m}, \ldots, z_{|C|,1}, \ldots, z_{|C|,m})$ be a solution vector to the VARIABLE-SIZED BIN COVERING problem. We restate LP (5.21) in the following form.

$$\lambda^* = \min\Big\{\lambda \mid \sum_{v\in\{0,\ldots,n\}^k} \frac{n(v,i)}{n(i)}\left(y_v + \sum_{j=1}^{m} z_{v,j}\right) \le \lambda \qquad 1 \le i \le k, x \in B_r \sum_{j=1}^{m}\sum_{v\in\widetilde{C}_j} \frac{r(v,j)}{p(\sigma_T)} z_{v,j} \le \lambda \qquad x \in B_r \Big\}, \tag{5.33}$$

where

$$B_r = \Big\{x \mid \sum_{j=1}^{m} d_j \left(\sum_{v\in C_j} y_v + \sum_{j=1}^{m}\sum_{v\in\widetilde{C}_j} z_{v,j}\right) = r\varepsilon \text{ and } \forall v, j : y_v \ge 0, z_{v,j} \ge 0\Big\} \tag{5.34}$$

LP (5.33) is a *convex block-angular resource sharing problem*. Note that for $\lambda = 1$ the constraints of LP (5.21) are equivalent to the constraints of LP (5.33). The value r defining the simplex B_r in (5.34) thereby will be set such that $r\varepsilon$ is the (approximate) value of an optimal solution and we can guess r via binary search. We explain this later in more detail. Suppose $r\varepsilon$ is the true value of an optimal integral solution. Then $\lambda^* = 1$ is the optimal value of the resource sharing problem and the corresponding solution vector x gives also a solution to LP (5.21). Jansen and Solis-Oba give in [42] an approximate solution to LP (5.33) for the case when $m = 1$ and $d_1 = 1$ and show how this can be transformed into

a $(1+\varepsilon)$-approximation for the VARIABLE-SIZED BIN COVERING problem. We can extend their technique to work for m bin types.

Resource sharing problems can be solved with the *price-directive decomposition method* [36,43] within any given approximation factor. This technique is also used by Jansen and Solis-Oba in [42]. We give a brief overview of this method.

An algorithm for solving a resource sharing problem finds a solution iteratively. It starts with an arbitrary feasible solution x^* and determines a price vector $c = (c_1, \ldots, c_{k+1})$ whose components are non-negative. It requires to solve a subproblem, called the *block program*, whose solution depends on c. We will state the block program for our problem below. A linear combination of an optimal solution $\hat{x}$ to the block program and the previous solution x^* found by the price-directive decomposition method so far determines an updated solution x^* for the original resource sharing problem. After a certain number of iterations for any given $\delta > 0$ the price-directive decomposition method guarantees a solution x^* with objective value at most $(1+\delta)\lambda^*$, where λ^* is the objective value of an optimal solution for the resource sharing problem. We are left to show how to transform this solution into a $(1+\varepsilon)$-approximate solution for the VARIABLE-SIZED BIN COVERING problem.

Statement of the Block Program

Let A denote the $(k+1) \times |C|(1+m)$ coefficient matrix corresponding to the constraints of LP (5.33). The price-directive decomposition method requires to solve approximately the block problem

$$\min\{\, cAx \mid x \in B_r \}. \tag{5.35}$$

Since B_r is a simplex, an optimal solution x^* for program (5.35) will be attained at a vertex. That is, one component of x^* has value $r\varepsilon$ and all other components are zero. It thus suffices to find the coordinate of x^* that has the smallest price, where we determine the price of a coordinate below.

Fix a variable y_v or $z_{v,j}$. Then we define the price of y_v and of $z_{v,j}$ to be $cAe_{j'}$, where j' is the column in the coefficient matrix A corresponding to y_v and $z_{v,j}$, respectively. Here $e_{j'}$ denotes the vector with a one in row j' and zero otherwise. Intuitively, the price of y_v and $z_{v,j}$ is obtained by multiplying the price vector c with the respective column j' of matrix A. By definition of A, we find that the price of y_v is $\sum_{i=1}^{k} n(v,i)c_i/n(i)$, and the price of $z_{v,j}$ is $\sum_{i=1}^{k} n(v,i)c_i/n(i) + r(v,j)c_{k+1}/p^+(\sigma_T)$.

Define two types of integer programs (IP)

$$\tau_{j,1} = \min \sum_{i=1}^{k} \frac{c_i}{n(i)} u_i \tag{5.36}$$

$$\text{subject to} \quad \sum_{i=1}^{k} s_i u_i \geq d_j$$

$$u_i \in \{0, \ldots, n(i)\}$$

and

$$\tau_{j,2} = \min \sum_{i=1}^{k} \frac{c_i}{n(i)} u_i + c_{k+1} \frac{d_j - \sum_{i=1}^{k} s_i u_i}{p(\sigma_T)} \tag{5.37}$$

$$\text{subject to} \quad \sum_{i=1}^{k} s_i u_i \leq d_j.$$

$$u_i \in \{0, \ldots, n(i)\}$$

Here the variables u_i denote the number of items of size s_i to choose. Hence it is not hard to see that IP (5.36) finds a cheapest configuration among those that cover a bin of type M_j and IP (5.37) finds a cheapest configuration among those that do not cover a bin of type M_j. Taking the overall cheapest configuration, i. e. the configuration which gives the minimum value in the set $\mathcal{M} = \{\tau_{j,1}, \tau_{j,2} \mid 1 \leq j \leq m\}$, is the configuration which is the solution to the block problem, and this configuration is uniquely associated to a variable y_v or $z_{v,j}$.

Solution of the Block Problem

In the previous section we reduced the problem of finding an optimal solution to the block problem to finding the configuration that gives the minimum value of the set $\mathcal{M}$. In this section we show how to find this minimum value by solving IPs (5.36) and (5.37).

IP (5.36) is the minimum knapsack problem and it is folklore that there exists a FPTAS for it. By a dynamic program and an appropriate rounding technique Jansen et al. [42] can also obtain an FPTAS for the program $\min\{\tau_{1,1}, \tau_{1,2}\}$, where $d_1 = 1$. We can use their FPTAS as a procedure in order to find the overall cheapest configuration, i. e. for m different and arbitrary d_j values. For this we scale the constraints appropriately:

Recall that $s = (s_1, \ldots, s_k)$ is the vector of all item sizes and vector $c = (c_1, \ldots, c_{k+1})$ is the price vector. Let $s(d_j) = (s_1/d_j, \ldots, s_k/d_j)$ and $c(d_j) = (c_1, \ldots, c_k, d_j c_{k+1})$. We replace the coefficients in the constraints of IPs (5.36)

and (5.37) by the corresponding coefficients from the vectors $s(d_j)$ and $c(d_j)$. We observe that $u = (u_1, \ldots, u_k)$ is a solution to the program

$$\tau'_{j,1} = \min \sum_{i=1}^{k} \frac{c_i}{n(i)} u_i \tag{5.38}$$

$$\text{subject to} \quad \sum_{i=1}^{k} \frac{s_i}{d_j} u_i \geq 1,$$

$$u_i \in \{0, \ldots, n(i)\}$$

if and only if u is a solution to IP (5.36). Note further that the respective objective values of the solutions are identical in both problems since all scaled values do not contribute to the objective function. Hence a solution u of IP (5.36) with objective value $\tau'_{j,1}$ is a solution of u of IP (5.38) with identical objective value.

Similarly we conclude that the program

$$\tau'_{j,2} = \min \sum_{i=1}^{k} \frac{c_i}{n(i)} u_i + d_j c_{k+1} \frac{1 - \sum_{i=1}^{k} u_i s_i / d_j}{p(\sigma_T)} \tag{5.39}$$

$$\text{subject to} \quad \sum_{i=1}^{k} \frac{s_i}{d_j} u_i \leq 1.$$

$$u_i \in \{0, \ldots, n(i)\}$$

has a solution u with objective value $\tau'_{j,2}$ if and only if u is a solution to IP (5.37) with the same objective value. Note that IP (5.38) and IP (5.39) are of the shape of IPs (5.36) and (5.37), where $d_j = 1$. Hence the FPTAS of Jansen and Solis-Oba for the block problem is applicable in this setting. Overall we have found an algorithm for solving the problem $\min\{\tau_{j,1}, \tau_{j,2}\}$: divide the item sizes in s by d_j and multiply the $(k+1)$-st component of the price vector c by d_j and compute a solution of the block problem with the modified size and price vector with the FPTAS of Jansen and Solis-Oba in [42].

As argued, the configuration minimizing $\min\{\tau_{j,1}, \tau_{j,2}\}$ over all bin types M_j, $j = 1, \ldots, m$, is a $(1+\varepsilon)$-approximate solution for the block problem of VARIABLE-SIZED BIN COVERING.

Approximation Guarantee and Running Time Analysis

Lemma 5.36. *A solution to LP (5.21) with objective value at least* $(1-2\varepsilon)\text{OPT} - O(1/\varepsilon^4)$ *and length* $O(1/\varepsilon^4)$ *can be found in polynomial time.*

Proof. As in the proof Theorem 5.29 we assume w. l. o. g. that $d_1 = 1$ and that the size of each item is smaller than 1. Then obviously $\text{OPT}(\mu, \sigma) \leq n$. Since we want to find an asymptotic FPTAS, we may assume that $\text{OPT}(\mu, \sigma) \geq 1$. We partition the interval $[1, n]$ into subintervals of size ε. Because $\text{OPT}(\mu, \sigma) \geq 1$, we know there exists a $\hat{r}$ such that $(1 - \varepsilon)\text{OPT}(\mu, \sigma) \leq \hat{r}\varepsilon \leq \text{OPT}(\mu, \sigma)$.

For given r let $\lambda(r)$ be the value of an optimal solution to LP (5.33) and $\lambda^*(r)$ be the value of the solution to LP (5.33) given by the price-directive decomposition method. If $r\varepsilon \leq \text{OPT}(\mu, \sigma)$, then $\lambda(r) \leq 1$, since $\lambda' = 1$ is the value of an optimal solution when $r\varepsilon = \text{OPT}(\mu, \sigma)$. In this case the price-directive decomposition method finds a solution with $\lambda^*(r) \leq 1 + \varepsilon$. If the price-directive decomposition method finds a solution with value $\lambda^*(r) > 1+\varepsilon$, we know that there is no solution with value $\lambda(r') \leq 1$ for any $r' \geq r$. Hence by binary search we find a largest r^* such that $\lambda(r^*) \leq 1 + \varepsilon$:

As argued, there exists an $\hat{r}$ such that $(1 - \varepsilon)\text{OPT}(\mu, \sigma) \leq \hat{r}\varepsilon \leq \text{OPT}(\mu, \sigma)$. Since for every $r' \leq \hat{r}$ a solution with value $\lambda(t') \leq 1 + \varepsilon$ can be found by the price-directive decomposition method, we find a $r^* \geq \hat{r}$. Thus $(1-\varepsilon)\text{OPT}(\mu, \sigma) \leq r^*\varepsilon$.

Also, the solution vector x^* corresponding to the solution with value $\lambda^*(r^*)$ may not be feasible, namely if $\lambda^*(r^*) > 1$. We can transform the solution vector x^* into a solution x' by multiplying each coordinate by the value $1 - \varepsilon$. It is easy to see that if x^* is a solution for LP (5.33) such that the left-hand side of each constraint has value $\lambda^* \leq 1 + \varepsilon$, then for the solution x' in LP (5.33) the left-hand side of each constraint has value at most $(1 - \varepsilon)\lambda^* \leq (1 - \varepsilon)(1 + \varepsilon) \leq 1 - \varepsilon^2 \leq 1$. Hence x' is a feasible solution for LP (5.21). As argued, the objective value $\lambda^*(r^*) \geq (1 - \varepsilon)$ and hence the objective value λ' for the scaled solution x' is at least $(1 - \varepsilon)^2 \geq 1 - 2\varepsilon$.

A solution x' may have up to $O((k + 1)(\varepsilon^{-2} + \ln(k + 1)))$ coordinates since there are so many calls to the block solver by the price-directive decomposition method [43]. We can transform this solution x' into a basic solution with at most $1 + 1/\varepsilon^4$ fractional coordinates in order to improve the approximation ratio. This can be done by solving a homogeneous linear system of equalities, see [42] for details.

We argue about the running time. The algorithm for a resource sharing problem with M constraints given by Jansen and Zhang [43] finds a solution in no more than $O(M(\varepsilon^{-2}+\ln M))$ iterations and has an overhead of $O(M \ln \ln(M/\varepsilon))$ operations per step. In our case it is $M = k + 1$.

The dynamic program of Jansen and Solis-Oba in [42], which we use as a procedure for solving the block problem, has a running time of $O(n^2/\varepsilon)$ per call. The overhead for scaling the price vector before we call this program is $O(k)$ and as we have m calls to this program we need time $O(m(k+n^2/\varepsilon))$ in order to solve the block problem. Note that, in particular, neither the running time of the block

solver nor of the algorithm from [43] depends on the size of $|B_r| = O(n^{1/\varepsilon^4})$.

We need an additional time of $O((k+1)(\varepsilon^{-2} + \ln(k+1))\mathcal{R}(2+k))$ for transforming the solution vector with $O((k+1)(\varepsilon^{-2} + \ln(k+1)))$ coordinates in an vector with $O(1+k)$ coordinates, where $\mathcal{R}(2+k)$ is the running time for solving a homogeneous linear system of $2+1/\varepsilon^4$ equations in $2+k$ variables. □

Proof. (*of Theorem 5.35).* The AFPTAS works identically as the APTAS, with the exception that we approximate LP (5.21) as given by Lemma 5.36. We first argue about the approximation guarantee. We can proceed as in the proof of Theorem 5.29. After Inequality (5.31) we have to take into account that LP (5.21) is only approximated. Then we can bound the additional loss of bins by the rounding procedure of the APTAS as done in Inequality (5.32). This gives then a feasible solution to the VARIABLE-SIZED BIN COVERING problem with an approximation guarantee at most $1+\varepsilon$ minus a constant number of bins. The running time is dominated by approximating LP (5.21) and hence given by Lemma 5.36. □

Chapter 6

Conclusions and Open Problems

In this thesis we have studied three semi-online makespan minimization problems and the GENERALIZED BIN COVERING problem. Several of our results are close to optimum: In Chapter 2, where we study makespan minimization with known total processing time, the gap between our lower bound of 1.585 in Theorem 2.6 and the best known upper bound of 1.6 [19] is small. In Chapter 3, where we study makespan minimization with job migrations, we obtain an algorithm with the best achievable competitive ratio using $o(n)$ migrations, cf. Theorems 3.3 and 3.7. Furthermore, Theorem 3.8 shows that the number of job migrations used by our algorithm can only be reduced by a small constant factor.

In Chapter 4 we have presented two algorithms for makespan minimization using parallel schedules. The number of schedules used by the $(4/3+\varepsilon)$-competitive algorithm in Theorem 4.5 is a constant and Theorems 4.16 and 4.17 show that this optimal upto a constant factor. Also the number of schedules used by the $(1+\varepsilon)$-competitive algorithm in Theorem 4.4 has to be a polynomial in $m^{\Omega(1/\varepsilon)}$.

In Chapter 5 we have presented a tight analysis of the algorithm NFD for VARIABLE-SIZED BIN COVERING in the unit supply model. The algorithm has an approximation ratio of $9/4 = 2.25$. Since the problem is inapproximable within a factor of 2, unless P = NP, our approximation guarantee is not far away from optimum, assuming P $\neq$ NP.

The obvious open questions are to determine the exact competitive ratio of the MINIMUM MAKESPAN problem when the total processing time of the jobs is given, to reduce the number of job migrations needed for our algorithms in Chapter 3 and to reduce the number of schedules needed by the algorithms in Chapter 4. For the latter problem in particular algorithms with a small constant independent of ε are interesting.

However, as the gaps in particular for the semi-online problems are small, further improvements might be hard to obtain. Hence, in semi-online makespan scheduling, a more fruitful working direction is probably to analyze the best com-

petitiveness attainable if an online scheduler knows the value of the optimum makespan. Note that this problem is closely related to the problem studied in Chapter 2. For the former, Azar and Regev [13] showed a lower bound of 4/3. The algorithm by Cheng et al. [19] is also 1.6-competitive in this problem setting.

The most interesting and challenging question is surely to tighten the gap between the best upper of [3] and lower bound of [18, 52] in ONLINE MINIMUM MAKESPAN. Even though we could not contribute directly to progress in this problem, we believe that at least the settings studied in Sections 2 and 4 lead to interesting structural insights, which could be useful in ONLINE MINIMUM MAKESPAN as well.

Concerning BIN COVERING it is maybe possible to increase the lower bound on the approximation guarantee of VARIABLE-SIZED BIN COVERING in the unit supply model. However, it seems more likely to us that the upper bound of 5 given by our algorithm in Theorem 5.3 can be decreased. We presume that a better upper bound has to treat the both cases of singular and regular coverage simultaneously, at least in the analysis. We believe that our analysis for the algorithm presented here is not tight. Regarding the infinite supply model of GENERALIZED BIN COVERING it is an interesting open question whether an AFPTAS can be obtained.

In the special case of VARIABLE-SIZED BIN COVERING we think it would be a good approach to modify the algorithm NFD in such a way that it tries to recombine left-over items with items that exceed a bin. A similar idea was used in [11] to improve asymptotic approximation ratios. We believe that in the unit supply model this idea could lead to an algorithm with an approximation guarantee of $2 + \varepsilon$, for a small $\varepsilon > 0$.

Bibliography

[1] Gagan Aggarwal, Rajeev Motwani, and An Zhu. The load rebalancing problem. *Journal of Algorithms*, 60(1):42–59, 2006.

[2] Susanne Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.

[3] Susanne Albers. On randomized online scheduling. In John H. Reif, editor, *Proc. on 34th Annual ACM Symposium on Theory of Computing*, pages 134–143. ACM, 2002.

[4] Susanne Albers and Matthias Hellwig. On the value of job migration in online makespan minimization. In Leah Epstein and Paolo Ferragina, editors, *Proc. 20th Annual European Symposium on Algorithms*, volume 7501 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2012.

[5] Susanne Albers and Matthias Hellwig. Semi-online scheduling revisited. *Theoretical Computer Science*, 443:1–9, 2012.

[6] Susanne Albers and Matthias Hellwig. Online makespan minimization with parallel schedules. *Submitted.* 2013.

[7] Enrico Angelelli, Á. B. Nagy, Maria Grazia Speranza, and Zsolt Tuza. The on-line multiprocessor scheduling problem with known sum of the tasks. *Journal of Scheduling*, 7(6):421–428, 2004.

[8] Enrico Angelelli, Maria G. Speranza, and Zsolt Tuza. Semi-on-line scheduling on two parallel processors with an upper bound on the items. *Algorithmica*, 37(4):243–262, 2003.

[9] Enrico Angelelli, Maria G. Speranza, and Zsolt Tuza. New bounds and algorithms for on-line scheduling: two identical processors, known sum and upper bound on the tasks. *Discrete Mathematics & Theoretical Computer Science*, 8(1):1–16, 2006.

[10] Enrico Angelelli, Maria G. Speranza, and Zsolt Tuza. Semi-online scheduling on two uniform processors. *Theoretical Computer Science*, 393(1-3):211–219, 2008.

[11] S. F. Assmann, David S. Johnson, Daniel J. Kleitman, and Joseph Y.-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5(4):502 – 525, 1984.

[12] Yossi Azar. On-line load balancing. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms*, volume 1442 of *Lecture Notes in Computer Science*, pages 178–195. Springer, 1996.

[13] Yossi Azar and Oded Regev. On-line bin-stretching. *Theoretical Computer Science*, 268(1):17–41, 2001.

[14] Yair Bartal, Amos Fiat, Howard J. Karloff, and Rakesh Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51(3):359–366, 1995.

[15] Yair Bartal, Howard J. Karloff, and Yuval Rabani. A better lower bound for on-line scheduling. *Information Processing Letters*, 50(3):113–116, 1994.

[16] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[17] E. Cesáro. Sur la série harmonique. *Nouvelles Annales de Mathématiques*, 3(4):295–296, 1885.

[18] Bo Chen, André van Vliet, and Gerhard J. Woeginger. A lower bound for randomized on-line scheduling algorithms. *Information Processing Letters*, 51(5):219–222, 1994.

[19] T. C. Edwin Cheng, Hans Kellerer, and Vladimir Kotov. Semi-on-line multiprocessor scheduling with given total processing time. *Theoretical Computer Science*, 337(1-3):134–146, 2005.

[20] Edward G. Coffman, Jr., Michael R. Garey, and David S. Johnson. Approximation algorithms for np-hard problems. chapter Approximation algorithms for bin packing: a survey, pages 46–93. PWS Publishing Co., Boston, MA, USA, 1997.

[21] János Csirik and Johannes B. G. Frenk. A dual version of bin packing. *Algorithms Review*, 1(2):87 – 95, 1990.

[22] János Csirik, Johannes B.G. Frenk, Martine Labbé, and Shuzhong Zhang. Two simple algorithms for bin covering. *Acta Cybernetica*, 14:13 – 25, 1999.

[23] János Csirik, David S. Johnson, and Claire Kenyon. Better approximation algorithms for bin covering. *Proc. 12th Symposium on Discrete Algorithms*, pages 557–566, 2001.

[24] János Csirik and V. Totik. Online algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, 21:163 – 167, 1988.

[25] János Csirik and Gerhard J. Woeginger. On-line packing and covering problems. In *Online Algorithms*, volume 1442 of *LNCS*, pages 147 – 177, 1998.

[26] Matthias Englert, Deniz Özmen, and Matthias Westermann. The power of reordering for online minimum makespan scheduling. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 603–612, 2008.

[27] Leah Epstein. Online variable sized covering. *Information and Computation*, 171(2):294– 305, 2001.

[28] Ulrich Faigle, Walter Kern, and György Turán. On the performance of online algorithms for partition problems. *Acta Cybernetica*, 9(2):107–119, 1989.

[29] William Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley & Sons, 1968.

[30] Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981.

[31] Rudolf Fleischer and Michaela Wahl. Online scheduling revisited. In Mike Paterson, editor, *8th Annual European Symposium on Algorithms*, volume 1879 of *Lecture Notes in Computer Science*, pages 202–210. Springer, 2000.

[32] Gábor Galambos and Gerhard J. Woeginger. An on-line scheduling heuristic with better worst case ratio than graham's list scheduling. *SIAM Journal on Computing*, 22(2):349–355, 1993.

[33] Todd Gormley, Nick Reingold, Eric Torng, and Jeffery Westbrook. Generating adversaries for request-answer games. In David B. Shmoys, editor, *Proc. 11th Symposium on Discrete Algorithms*, pages 564–565. ACM/SIAM, 2000.

[34] Ronald L. Graham. Bounds for certain multi-processing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.

[35] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[36] Michael D. Grigoriadis and Leonid G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Mathematics of Operations Research*, 21:321–340, May 1996.

[37] Matthias Hellwig and Alexander Souza. Approximation algorithms for generalized and variable-sized bin covering. In Anupam Gupta, Klaus Jansen, José D. P. Rolim, and Rocco A. Servedio, editors, *APPROX-RANDOM*, volume 7408 of *Lecture Notes in Computer Science*, pages 194–205. Springer, 2012.

[38] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM*, 34:144–162, 1987.

[39] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[40] John F. Rudin III. *Improved bounds for the on-line scheduling problem*. PhD thesis, The University of Texas at Dallas, May 2001.

[41] John F. Rudin III and Ramaswamy Chandrasekaran. Improved bounds for the online scheduling problem. *SIAM Journal on Computing*, 32(3):717–735, 2003.

[42] Klaus Jansen and Roberto Solis-Oba. An asymptotic fully polynomial time approximation scheme for bin covering. *Theoretical Computer Science*, 306(1 – 3):543 – 551, 2003.

[43] Klaus Jansen and Hu Zhang. Approximation algorithms for general packing problems with modified logarithmic potential function. In *Proc. 2nd International Conference on Theoretical Computer Science, Montreal, Canada*, pages 255–266, 2002.

[44] David R. Karger, Steven J. Phillips, and Eric Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 20(2):400–430, 1996.

[45] Hans Kellerer, Vladimir Kotov, Maria Grazia Speranza, and Zsolt Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

[46] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979.

[47] Ming Liu, Yinfeng Xu, Chengbin Chu, and Feifeng Zheng. Online scheduling on two uniform machines to minimize the makespan. *Theoretical Computer Science*, 410(21-23):2099–2109, May 2009.

[48] Frank D. Murgolo. An efficient approximation scheme for variable-sized bin packing. *SIAM Journal on Computing*, 16(1):149–161, 1987.

[49] Peter Sanders, Naveen Sivadasan, and Martin Skutella. Online scheduling with bounded migration. *Mathematics of Operations Research*, 34(2):481–498, 2009.

[50] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999.

[51] Steven S. Seiden. Online randomized multiprocessor scheduling. *Algorithmica*, 28(2):173–216, 2000.

[52] Jiří Sgall. A lower bound for randomized on-line multiprocessor scheduling. *Information Processing Letters*, 63(1):51–55, 1997.

[53] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.

[54] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Communication of the ACM*, 28(2):202–208, 1985.

[55] Zhiyi Tan and Shaohua Yu. Online scheduling with reassignment. *Operations Research Letters*, 36(2):250–254, 2008.

[56] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2004.

[57] Ingo Wegener. *Complexity theory - exploring the limits of efficient algorithms*. Springer, 2005.

[58] Gerhard J. Woeginger and Guochuan Zhang. Optimal on-line algorithms for variable-sized bin covering. *Operations Research Letters*, 25(1):47 – 50, 1999.

Erklärung

Hiermit erkläre ich,

- dass ich die vorliegende Dissertationsschrift „On Semi-Online Machine Scheduling and Generalized Bin Covering“ selbstständig und ohne unerlaubte Hilfe angefertigt habe,
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze, und
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist, gemäß amtlichem Mitteilungsblatt Nr. 34/2006.

Berlin, den ....